République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE d'ORAN Mohamed Boudiaf

Laboratoire LSSD

Faculté des Sciences
Département d'informatique
Spécialité :Informatique. Option : Systèmes, Réseaux et Bases De Données

MEMOIRE
Présenté par

Mr. MOUNA Azzeddine

Pour l'obtention du diplôme de Magister en Informatique
Thème

DATAMINING DISTRIBUE DANS LES GRILLES : APPROCHE REGLES D'ASSOCIATION

Devant la commission d'examen composée de :

Qualité	**Nom et Prénoms**	**Grade**	**Etb d'origine**
Président	MR CHOUARFIA Abdellah	M.conf. A	USTOMB
Rapporteur	Mme BELBACHIR Hafida	Professeur	USTOMB
Examinateur	MR DJEBBAR Bachir	Professeur	USTOMB
Examinateur	Mme ZAOUI Linda	M.conf. A	USTOMB
Invitée	Mme MEKEDDEM Djamila	chargée de cours	USTOMB

Année universitaire : 2012/2013

REMERCIEMENTS

Ce projet n'aurait pas abouti sans la bénédiction de Dieu le tous puissant, Qui m'a donné le courage et la patience pour mener à bien ce travail.

Je tiens à remercier toutes les personnes qui ont contribué de manière directe ou indirecte à l'aboutissement de ce travail.

En premier lieu, J'exprime ma gratitude à Madame H. Belbachir mon encadreur, qui m'a confié ce sujet d'actualité. J'apprécie son enthousiasme, sa gentillesse, ses conseilles et ses orientations remarquables.

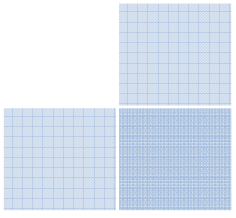
Je tiens à remercier également tous les membres du jury, Mr CHOUARFIA autant que président, Mme ZAOUI et Mr DJABBAR comme examinateurs qui ont accepté de juger ce travail et à remercier Mme MOKADDEM d'avoir accepté notre invitation.

Merci aux membres du laboratoire de Base de données, à mes amis et à tous les enseignants pour leurs efforts tout au long de mes années d'études.

Enfin, un grand Merci à tous mes proches qui ont apporté leurs soutien durant l'élaboration de ce travail.

TABLE DES MATIÈRES

Chapitre III – proposition pour le problème des Règles d'associations distribuées sur les Grilles



LISTE DES FIGURES

Chapitre I- Extraction des Règles d'Associations : Problème et Approche

Chapitre II- Contribution au problème d'extraction des Règles d'associations distribuées

Chapitre III – Proposition pour le problème des Règles d'associations distribuées sur les Grilles

Introduction générale

Le DataMining est une discipline qui se base essentiellement sur les pratiques et les expériences vécues par des entreprises qui ont décidé d'investir dans ce domaine. Le DataMining peut être vu comme la formulation, l'analyse et l'implémentation d'un processus de transformation de données en des connaissances. Les techniques de DataMining sont de plus en plus employées dans des domaines scientifiques et dans des domaines industriels. Les données impliquées dans le DataMining sont souvent caractérisées non seulement par leurs volume important (un grand nombre d'enregistrements) mais aussi par leurs grandes dimensions (un grand nombre d'attributs). Le DataMining est souvent utilisé pour traiter des données complexes comme des images, des données géographiques, des données scientifiques, des documents non structurés ou semi-structurés, etc. À la base, les données peuvent être de n'import quel type.

Les outils classiques de DataMining s'exécutent souvent sur les données centralisées dans un seul site. Cependant, durant ces dernières années, les entreprises sont témoins d'une grande Explosion de l'information[1], d'où la nécessité d'utiliser des outils de haute performance pour le passage à l'échelle et l'évolutivité des méthodes sous-jacentes.

Le DataMining haute performance est une évolution naturelle des technologies de ce type d'outil, motivé par le besoin des systèmes scalables et haut performants pour un passage à l'échelle des quantités massives de données manipulés qui ne cessent d'augmenter et la puissance de calcul induit par le traitement de ces données.

Initialement, des systèmes parallèles sont exploités pour l'exécution des techniques de DataMining (**DataMining Parallèle)**. Le parallélisme, et comme d'autres applications intensives aux données, est purement employé dans le but d'accélérer et de réduire le temps du traitement. Les systèmes parallèles avec une architecture partagée/distribuée ont beaucoup de propriétés désirables tel que l'utilisation simultanée de plusieurs processeurs et l'existence d'une mémoire commune ou partagée par les processeurs. Les programmes parallèles sont faciles à mettre en œuvre sur de tels systèmes. Ces derniers se caractérisent par la disponibilité d'un réseau d'interconnexion interne rapide et l'homogénéité d'un système d'exploitation unique. Cependant, la limite de la largeur de la bande du bus commun peut limiter la scalabilité de ces systèmes. En plus, le problème de ces systèmes réside dans leur coût, les rendant ainsi, indisponibles en dehors de sociétés et de centres scientifiques. Par conséquent, une nouvelle tendance est apparue et les chercheurs sont orientés au **DataMining Distribuée** comme un processus de DataMining opérant par une analyse des données d'une façon distribuée [KAR 00]. La distribution des données et des calculs permet de résoudre des problèmes plus complexes et d'exécuter des applications de natures distribuées. Contrairement au modèle centralisé, l'approche distribuée suppose que les

[1] C'est un terme qui décrit la croissance rapide de la quantité d'information publiée et les effets de cette abondance de données.

sources de données sont distribuées à travers des sites multiples. Les algorithmes développés dans ce gisement adressent le problème de l'obtention efficace des résultats d'extractions à partir de toutes les données à travers ces sources distribuées. Une approche simple pour le DataMining Distribué consiste à appliquer les outils de DataMining existants indépendamment sur chaque site et de combiner les résultats. Cependant, cela ne donne pas souvent des résultats globaux valides. Par conséquent L'approche utilisée est la fouille de données de façon distribuée à travers la coopération des différents sites du système par des communications/synchronisations et la fusion des résultats locaux entre chaque étape.

La technologie de Grilles peut intégrer à la fois les systèmes parallèles et distribués, et elle représente une infrastructure critique pour le DataMining distribué [TAL 06]. Ainsi, une nouvelle tendance de DataMining est apparue, on parle de DataMining sur les Grilles. Les Grilles sont des infrastructures de calcul distribuées qui rend possible la coordination des ressources partagées dans des organisations dynamiques constituées d'individus, d'institutions et de ressources. Les Grilles étendent les paradigmes de calcul parallèle et distribué par la négociation et l'allocation dynamique des ressources hétérogènes, l'utilisation des protocoles et des services standards ouverts. Nous nous intéressons dans notre travail aux problèmes des règles d'associations distribuées dans les Grilles. Les règles d'associations sont de puissantes méthodes qui ont comme objectif de trouver les régularités dans les tendances des données. Les règles d'associations essayent de trouver l'ensemble de transactions qui apparaît souvent ensemble et qui sont exprimés sous forme de règles. Le problème d'extraction de règles d'associations nécessite un accès à la totalité des données a traité. La méthode s'appuie sur des mesures globaux (support, confiance,…etc) pour générer les règles d'association les plus pertinentes et de réduire l'espace de stockage.

L'extraction des Itemsets fréquents dans les bases de données est l'étape primordiale et la plus coûteuse dans le processus d'extraction des règles d'association. L'accent est mis dans les travaux de recherche sur cette étape qui représente le principal coût de traitement dans le cas de grandes bases de données, à cause de la génération des candidats. Cette phase est la plus coûteuse du fait de la taille exponentielle de l'espace de recherche et du nombre élevé de balayages de la totalité de la base de données ; ce balayage est nécessaire pour évaluer les supports des Itemsets et de calculer par la suite les confiances des règles générées. Ces deux critères (nombre de balayages et nombre d'Itemsets générés) sont les deux facteurs d'efficacité ou d'inefficacité d'un algorithme d'extraction des Itemsets fréquents.

Néanmoins, le recours au calcul haut performant parallèle et distribué exige que le problème des règles d'associations doit être reformulé et réadapté aux besoins et spécificités des environnements de calcul haut performant. L'utilisation de la méthode d'extraction des règles d'associations dans un contexte distribué consiste à effectuer les traitements sur des vues partielles de données disponibles localement sur chaque site ce qui nécessite un grand nombre de

communications afin d'obtenir des résultats valides sur l'ensemble global des données. Le traitement suppose de comparer chaque partie de la base de données à toutes les autres parties (haute complexité). Ces communications sont soumises à des synchronisations, entre chaque itération des algorithmes, qui sont très pénalisantes dans un environnement distribué et particulièrement dans les Grilles.

Notre travail concerne l'étude du problème de la recherche des règles d'associations distribuées sur les Grilles dans le but de proposer de nouvelles solutions et de concevoir de nouveaux algorithmes (heuristiques) bien adaptés aux environnements de Grilles. L'accent est mis essentiellement sur l'amélioration des performances de l'étape d'extraction des Itemsets fréquents (la phase la plus coûteuse dans le processus de recherche des règles d'associations) par la proposition d'une nouvelle approche qui se base sur la notion de Support$^+$ que nous avons introduite. Cette approche permet la réduction du nombre d'Itemsets candidats générés par les algorithmes d'extraction des Itemsets fréquents en modifiant la phase de génération-élagage des candidats dans ces algorithmes. De plus, notre approche permet la réduction du nombre de parcours de base de données nécessaires pour le calcul des supports des Itemsets, et par conséquence le nombre de communications et/ou synchronisations nécessaires pour le calcul de l'information globale dans un contexte distribué. Nous avons proposé des stratégies plus scalables et plus adaptées de l'utilisation de notre méthode dans un environnement distribué en adoptant une stratégie qui permet de réduire le nombre de communications/synchronisations entre les différents sites et qui permet d'éviter les calculs redondants pour la génération des Itemsets candidats. En fin, nous avons proposé un scénario typique de déploiement et d'utilisation de notre méthode dans un environnement Grille par la proposition d'une version adaptée à l'utilisation à grande échelle en examinant les possibilités d'exposer notre méthode sous forme d'un Service Grille.

Ce document est composé de trois chapitres :

- Dans le premier chapitre, nous présentons la définition, les tâches et les techniques de DataMining. Nous présentons par la suite les concepts et les notions de bases de la technique des règles d'associations et le processus complet de leurs extractions. Nous détaillerons ensuite le problème d'extraction des Itemsets fréquents et nous présenterons à la fin le principe général de notre approche pour l'extraction des Itemsets fréquents et les expérimentations effectuées dans un contexte séquentiel pour l'évaluation de notre méthode.

- Dans le deuxième chapitre, nous abordons le problème des règles d'associations distribuées en particulier le problème d'extraction des Itemsets fréquents distribuées en présentant une étude approfondie sur les algorithmes distribués d'extractions d'Itemsets fréquents existants et en soulignant les problèmes et les limites de ces algorithmes. Nous présentons dans la dernière section de ce chapitre notre contribution et les deux algorithmes distribués d'extractions

d'Itemsets fréquents que nous avons introduits et les expérimentations effectuées pour l'évaluation de notre approche distribuée.

-Dans le troisième chapitre nous abordons le problème des règles d'associations sur les plateformes de Grilles en présentant premièrement les définitions et les concepts relatifs aux Grilles informatiques et deuxièmement les principaux environnements Grilles dédiés aux techniques de DataMining. Nous présentons par la suite notre étude faite sur les principaux travaux d'implémentation/déploiement de la technique des règles d'associations sur les Grilles. Nous présentons dans la dernière section de ce chapitre notre proposition pour le problème d'extraction des règles d'association distribuée dans un environnement Grille en explorant la possibilité d'exposer notre méthode sous forme d'un service Grille.

Enfin, une conclusion et des perspectives de notre travail clôturent ce document.

1 Introduction :

Les évolutions technologiques réalisées ces dernières années ont permis de diminuer de façon considérable les coûts de collecte et de stockage de données. De nombreuses entreprises ont saisi l'opportunité d'archiver de nombreuses informations (informations clients, informations produits, …) dans l'espoir de pouvoir les exploiter. Ces sources d'informations potentielles jouent un rôle important dans le domaine de la concurrence ; il reste à les utiliser au mieux.

Les techniques traditionnelles de statistiques ne sont pas utilisables sur d'aussi grandes quantités de données sans l'utilisation de techniques d'échantillonnage. Le fDataMining apparait alors comme la solution pour extirper des connaissances de cet amas de données disponibles.

Le DataMining répond à un ensemble de tâches du domaine scientifique en fournissant un ensemble de technique qui répond à ces tâches spécifiques. L'extraction de règle d'association est devenue une des tâches fondamentales du DataMining, Les règles d'association sont les méthodes les plus répandues dans le domaine de marketing et de la distribution. Leur principe consiste à trouver les groupes d'articles qui apparaissent le plus fréquemment ensembles, et de générer des règles d'association. Ces règles sont simples, faciles à comprendre et assorties d'une probabilité, ce qui en fait un outil agréable et directement exploitable par l'utilisateur métier.

Bien qu'elle soit traditionnellement liée au secteur de la distribution, l'extraction des règles d'association peut s'appliquer à d'autres domaines :

- L'utilisation d'une carte de crédit (location de voiture, réservation de chambre d'hôtel) peut renseigner sur les habitudes d'un client.
- Les options achetées par les clients des télécommunications (attente d'appel, renvoi d'appel, appel rapide, etc.) aident à savoir comment regrouper ces services pour maximiser le revenu.
- Les carnets de santé des patients peuvent donner des indications sur les complications induites par certaines combinaisons de traitements,
- Les services bancaires (comptes numéraires, bons de caisse, services d'investissement, locations de voitures) permettent d'identifier les clients susceptibles de désirer d'autres services.
- La combinaison dans les demandes d'assurances inhabituelles peut signaler une fraude et déclencher des suppléments d'enquête.

Dans ce chapitre, nous présentons le processus général d'extraction des règles d'associations. Ensuite, nous abordons en détail le problème d'extraction des Itemsets fréquents. Nous présenterons à la fin le principe général de notre approche pour l'extraction des Itemsets fréquents et les expérimentations effectuées.

2 Le DataMining

Le terme de DataMining est souvent employé pour désigner l'ensemble des outils permettant à l'utilisateur d'accéder aux données de l'entreprise et de les analyser.

On distingue le terme « DataMining » (fouille de données) et le terme « Extraction de connaissance» [PIA 91], même si par abus de langage ces termes sont utilisés pour définir la découverte (ou extraction) de connaissance : KDD (Knowledge Discovery in Databases). L'extraction de connaissances est réalisée grâce au processus particulier qu'est la fouille de données (DataMining) qui apparaît donc comme une méthode d'extraction parmi d'autres.

Nous restreindrons ici le terme de DataMining aux outils ayant pour objet de générer des informations riches à partir des données de l'entreprise, notamment des données historiques, de découvrir des modèles implicites dans les données.

On pourrait définir le DataMining comme une démarche d'exploration de données ayant pour objet de découvrir des relations et des faits (des connaissances), à la fois nouveaux et significatifs sur de grands ensembles de données.

Ces concepts s'appuient sur le constat qu'il existe au sein de chaque entreprise des informations cachées dans le gisement de données. Ils permettent, grâce à un certain nombre de techniques spécifiques, de faire apparaître des connaissances. [CNA 98].

2.1 Les tâches du DataMining :

Une multitude de problèmes d'ordre intellectuel, économique ou commercial peuvent être regroupés, dans leur formalisation, dans l'une des tâches suivantes :

- **La classification :** La classification consiste à examiner des caractéristiques d'un élément nouvellement présenté afin de l'affecter à une classe d'un ensemble prédéfini [BER 96].
- **L'estimation** : Contrairement à la classification, le résultat d'une estimation permet d'obtenir une variable continue. Celle-ci est obtenue par une ou plusieurs fonctions combinant les données en entrée. Le résultat d'une estimation permet de procéder aux classifications grâce à un barème.
- **La prédiction** : La prédiction ressemble à la classification et à l'estimation mais dans une échelle temporelle différente. Tout comme les tâches précédentes, elle s'appuie sur le passé et le présent mais son résultat se situe dans un futur généralement précisé.
- **La Segmentation** : La segmentation consiste à segmenter une population hétérogène en sous populations homogènes. Contrairement à la classification, les sous-populations ne sont pas préétablies.
- **La description :** C'est souvent l'une des premières tâches demandées à un outil de DataMining. On lui demande de décrire les données d'une base complexe. Cela engendre souvent une exploitation supplémentaire en vue de fournir des explications.

- **L'optimisation :** Pour résoudre de nombreux problèmes, il est courant pour chaque solution potentielle d'y associer une fonction d'évaluation. Le but de l'optimisation est de maximiser ou minimiser cette fonction. Quelques spécialistes considèrent que ce type de problème ne relève pas du DataMining.

2.2 Les techniques de DataMining :

Plusieurs techniques peuvent être inscrites dans le contexte du DataMining, on en cite :

- **Les arbres de décision :** ce sont des outils très puissants principalement utilisés pour la classification, la description ou l'estimation. Il s'agit d'une représentation graphique sous forme d'arbre de décision représentant un enchaînement hiérarchique de règles logiques qui permet de diviser la base d'exemples en sous-groupes.
- **Les réseaux de neurones :** inspirés de la biologie, les réseaux de neurones représentent une transposition simplifiée des neurones du cerveau humain. Ils sont utilisés dans la prédiction et la classification.
- **Les algorithmes génétiques** : ce sont des méthodes d'optimisation de fonctions basée sur les mécanismes génétiques pour élaborer des paramètres de prévision des plus optimaux. Ils permettent de résoudre des problèmes divers, notamment d'optimisation, d'affectation ou de prédiction.
- **Les règles d'association** : Ce sont les méthodes les plus répandues dans le domaine de marketing et de la distribution. Elles peuvent être appliquées dans différents secteurs d'activité pour lesquels, il est intéressant de trouver des groupements d'articles qui apparaissent le plus fréquemment ensembles, et de générer des règles d'associations. Le but principal de cette technique est descriptif ou prédictif.
- **Les plus proches voisins (CBR, Case Based Reasoning) :** C'est une méthode dédiée à la classification qui peut être étendue à des tâches d'estimation. De façon similaire, le « raisonnement basé sur la mémoire », technique du DataMining dirigé, permet de classer ou de prédire des données inconnues à partir d'instances connues.
- **Le Clustering :** C'est une classification non supervisée, les classes possibles et leur nombre ne sont pas connus à l'avance et les exemples disponibles sont non étiquetés. Le but est donc de découvrir des relations intéressantes qui peuvent exister implicitement entre les données et qui permettront de regrouper dans un même groupe ou cluster les objets considérés comme similaires.

Nous nous intéresserons dans nos travaux à la technique des règles d'associations. Nous détaillerons plus loin dans ce chapitre les grandes lignes de cette technique.

3 La technique d'extraction des règles d'associations

L'extraction des règles associatives est une méthode qui a vu le jour avec la recherche en bases de données, initialement introduite par Agrawal [AGR 93] pour l'analyse du panier de la ménagère pour retrouver les relations entre produits. Elle permet d'analyser les tickets de caisse des clients afin de comprendre leurs habitudes de consommations, agencer les rayons du magasin, organiser les promotions, gérer les stocks, etc.

Dans les bases de données de ventes, un enregistrement consiste en une transaction regroupant l'ensemble des articles achetés appelées items. Ainsi une base de données est un ensemble de transactions, qu'on appelle aussi **base transactionnelle** ou **base de transactions**.

Par exemple on sera capable de dire que 70% des clients qui achètent du lait achètent en même temps des œufs (lait → œuf : 0.70), une telle constatation est très intéressante puisqu'elle aide le gestionnaire d'un supermarché à ranger ses rayons de telle sorte que le lait et les œufs soient à proximité.

3.1 Concepts et définitions:

Nous rappelons brièvement les notions de base dans ce cadre théorique :

- Contexte de fouille :

Un contexte de fouille est un triplet $\beta = (O, A, R)$ décrivant un ensemble fini **O** d'objets (ou de transactions), un ensemble fini **A** d'attributs (ou items) et une relation (d'incidence) binaire **R** ($R \subset O \times A$). Chaque couple $(o, a) \in R$, désigne le fait que l'objet $o \in O$, possède l'attribut $a \in A$.

Pour instance, nous prenons l'exemple de contexte de fouille cité dans [SALL 03] : Un complexe cinématographique a décidé de fidéliser son public en lançant la carte d'abonnement au cinéma dit "illimité". Les films vus par chaque cinéphile sont enregistrés dans une base de données à chaque fois que le client se présente au guichet.

Elle est exploitée par la suite pour comprendre les attitudes de "consommation" du cinéma, les types de films les plus prisés par le public, les heures auxquelles les gens préfèrent venir voir un film, etc.

La table β ci-dessous est un extrait (fictif) et donne pour chaque cinéphile identifié par un numéro tid, l'ensemble des films qu'il a vus durant le mois courant. Les films concernés sont donnés dans la table *I*.

Par exemple la ligne d'identificateur tid=1 de β concerne un client ayant vu dans le mois les trois films suivants : « Harry Potter », « Attrape moi si tu peux» et « Un homme d'exception ».

I		
Item	**Titre**	**Réalisateur**
A	Harry Potter	Chris Columbus
B	Star Wars II	George Lucas
C	Attrape-moi si tu peux	Steven Spielberg
D	Un homme d'exception	Ron Howard
E	Taken	Steven Spielberg

β	
Identifiant	**Transaction**
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

Fig. 1.1 Exemple de contexte de fouille

- Item :

Un item est tout article, attribut, littéral, appartenant à un ensemble fini d'éléments distincts I = (x_1, x_2,…, x_n).

L'ensemble I contient les items A, B, C, D, E correspondant aux films projetés pendant une période dans une salle de cinéma.

- Itemset :

On appelle Itemset tout sous-ensemble d'items de I. Un Itemset constitué de k items sera appelé un k-Itemset.

Pour simplifier, on écrira un Itemset sans les accolades et sans les virgules séparant les éléments de l'ensemble. L'Itemset {A, B, C} est un 3-Itemset noté ABC.

- Ordre sur les items :

L'ensemble des items I étant fini, on choisit une application h : I $\rightarrow$ {1,…, | I |} qui associe à chaque item x $\in$ I un entier naturel.

On a alors un ordre total noté « < » sur les items de I. On notera l'ensemble I muni de l'ordre « < » entre items de I par (I, <).

- Treillis :

Un ensemble ordonné (T, $\leq$) est un treillis si toute paire d'éléments de T possède une borne inférieure et une borne supérieure.

On notera la borne inférieure de T par $\perp$ et la borne supérieure par $\top$.

- Treillis (P (I),$\subseteq$)):

L'ensemble P (I) des parties d'un ensemble I muni de l'inclusion $\subseteq$ est un treillis.

Les opérations binaires $\wedge$ et $\vee$ sont respectivement $\cap$ et $\cup$. De plus, ce treillis admet une borne inférieure $\perp = \phi$ et une borne supérieure $\top$ = I.

La figure 1.2 représente le treillis des Itemsets pour I = {A, B, C, D, E}. On retrouve au niveau 2 du treillis, tous les Itemsets de taille 2, dite 2-Itemsets.

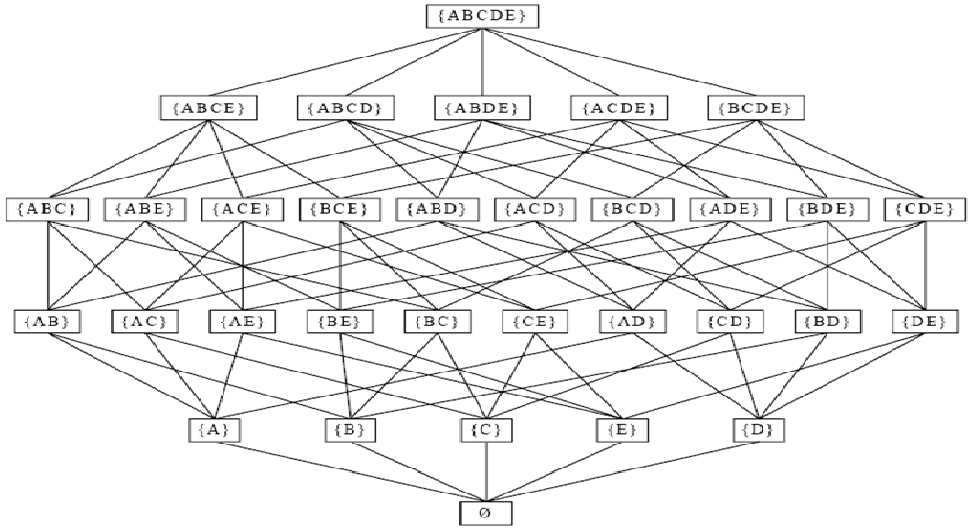


Fig. 1.2 Treillis des Itemsets pour *I* = {A, B, C, D, E}

-Transaction :

Une transaction est un Itemset identifié par un identificateur unique tid. L'ensemble de tous les identificateurs de transactions tids sera désigné par l'ensemble T.

Par exemple l'ensemble des produits achetés par un client dans un supermarché représente une transaction, l'ensemble des films vus par un cinéphile représente aussi une transaction.

- Fréquence:

La fréquence d'un Itemset X, noté freq(X), est le nombre de transactions du contexte β contenant X :

$$Freq(X) = |\ \{(y, X_y) \in \beta / X \subseteq X_y\}\ |$$

Dans l'exemple de la figure 1.1, on a Freq(AB) = 2, vu que l'Itemset AB apparaît dans les transactions 3 et 5 de β.

- *Support* :

Le *Support* d'un Itemset X, noté Supp (X) est la proportion de transactions de β contenant X:

$$Supp\ (X) = |\ \{(y, Xy) \in \beta / X \subseteq Xy\}|\ /\ |\beta|$$

Le *Support* prend sa valeur dans l'intervalle [0,1].

Dans l'exemple de la figure 1.1, on a Supp(AB)= 0,33 (33%) vu que l'Itemset AB apparaît dans deux transactions parmi 6 de β.

- Itemset fréquent:

Étant donné un seuil μ, appelé *Support minimum* (*Minsup*), un Itemset X est dit fréquent (relativement à μ) dans une base de transactions β, si son *Support* dépasse un seuil fixé à priori *Minsup*:

$$X \text{ est fréquent si et seulement si } Supp(X) \geq \mu$$

Dans l'exemple de la figure 1.1, pour un *Support* de μ = 33%, l'Itemset AC de *Support* égal à 3/6 = 50% est fréquent.

- Propriété d'antimonotonicité :

Tout sous-ensemble d'un Itemset fréquent est un Itemset fréquent.

Intuitivement, cela veut dire que si un Itemset a un *Support* inférieur au *Support* minimum aucun de ses sur-ensembles n'aura un *Support* supérieur ou égal au *Support minimum*.

- Bordure positive :

On appelle bordure positive B^+ l'ensemble des Itemsets fréquents dont tous les sur-ensembles sont peu fréquents. Ces Itemsets correspondent aux Itemsets fréquents maximaux.

$$B^+ = \{X \subseteq I \ / \ Supp(X) \geq \mu \ , \ \forall \ X' \supseteq X, \ Supp(X') < \mu\}$$

- Bordure négative :

On appelle bordure négative B^- l'ensemble des Itemsets peu fréquents dont tous les sous-ensembles sont fréquents. Ces Itemsets correspondent aux Itemsets peu fréquents minimaux.

$$B^- = \{X \subseteq I \ / \ Supp(X) < \mu \ , \ \forall \ X' \subseteq X, \ Supp(X') \geq \mu\}$$

- Une règle d'association :

Soit X un Itemset et A un sous-ensemble de X. Une règle d'association est une règle de la forme $A \rightarrow X - A$, exprimant le fait que les items de A tendent à apparaître avec ceux de $X - A$.

A s'appelle l'antécédent de la règle et $X - A$ le conséquent de la règle noté C.

Par exemple la règle d'association $A \rightarrow B$ exprime le fait que les cinéphiles ayant vu « Harry Potter » tendent aussi à voir « Star Wars ».

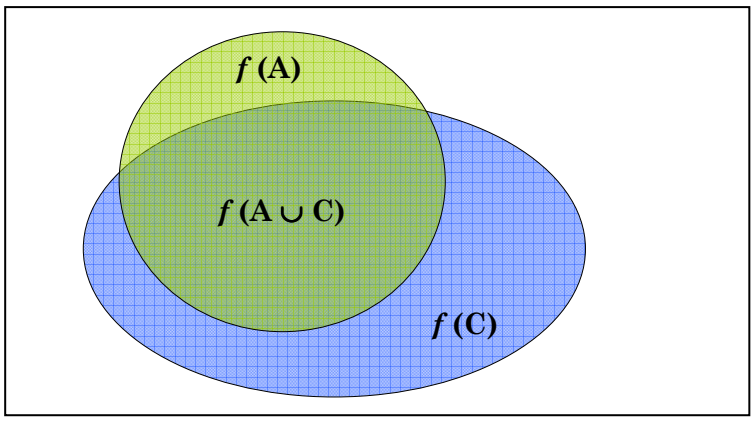


Fig. 1.3 Inclusion des transactions contenant X.

NB : Pour un Itemset X, *f*(X) représente l'ensemble des transactions contenant X (tidset).

- Support d'une règle :

Le *Support* d'une règle d'association A → C est le nombre de transactions contenant à la fois tous les items de A et tous les items de C, par rapport au nombre total de transactions:

$$\text{Supp}\ (\boldsymbol{A} \rightarrow \boldsymbol{C}) = |\ f\ (\boldsymbol{A} \cup \boldsymbol{C})\ |\ /\ |\ \beta\ |$$

- Confiance d'une règle :

La confiance d'une règle d'association A → C, noté Conf (A → C) représente le nombre de transactions contenant à la fois tous les items de A et tous les items de C, par rapport au nombre de transactions contenant les items de A. On peut écrire aussi :

$$\text{Conf}\ (\boldsymbol{A} \rightarrow \boldsymbol{C}) = |\ f\ (\boldsymbol{A} \cup \boldsymbol{C})\ |\ /\ |\ f\ (\boldsymbol{A})| = \text{Supp}\ (\boldsymbol{A} \cup \boldsymbol{C})\ /\ \text{Supp}\ (\boldsymbol{A})$$

$$\text{Avec} : 0 \leq \text{Conf}\ (\boldsymbol{A} \rightarrow \boldsymbol{C}) \leq 1.$$

La figure 1.3 illustre la notion de confiance. On a deux ensembles de transactions, celles couvrant A et celles couvrant C, à savoir f (A) et f (C) respectivement.

Nous savons que f (A) ∩ f (C) = f (A ∪ C).

La confiance mesure le degré d'inclusion de A dans C. C'est une mesure permettant d'évaluer la solidité d'une règle d'association.

- Règle d'association solide :

Une règle d'association A → C est dite solide si étant donné un Support minimum μ, l'Itemset A ∪ C est fréquent et si sa confiance dépasse un seuil donné, fixé a priori, appelé le seuil de confiance minimum (*Minconf*). Dans la suite il sera noté λ .

$\boldsymbol{A} \rightarrow \boldsymbol{C}$ est solide si et seulement si $\text{Supp}\ (\boldsymbol{A} \cup \boldsymbol{C}) \geq \mu$ et $\text{Conf}\ (\boldsymbol{A} \rightarrow \boldsymbol{C}) \geq \lambda$.

Par exemple si on prend λ = 33 % et μ = 33 %, la règle d'association A → C est considérée comme une règle solide car sa confiance, égale à 100%, dépasse le seuil de confiance minimum λ = 33 % et son *Support* de 50% dépasse le Support minimum de μ = 33 %.

Plus le *Support* d'une règle est élevé, plus la règle est fréquente. Plus sa confiance est élevée, moins il y a de contre-exemples de cette règle.

3.2 Processus d'extraction des règles d'association

L'extraction des règles d'association peut être décomposée en quatre étapes représentées dans la figure suivante :

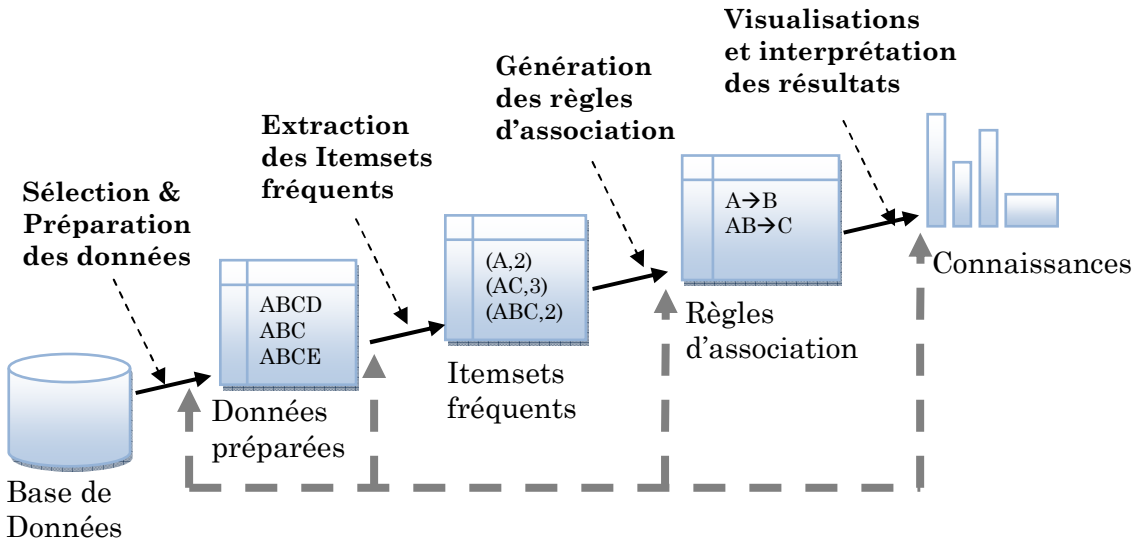


Fig. 1.4 Processus d'extraction des règles d'association

- **Sélection et préparation des données**

Cette étape permet de préparer les données afin de leur appliquer les algorithmes d'extraction des règles d'association. Elle est constituée de deux phases :

- La sélection des données de la base qui permettront d'extraire les informations intéressant l'utilisateur. Ainsi la taille des données traitées est réduite ce qui assure une meilleure efficacité de l'extraction.
- . La transformation de ces données en base de données transactionnelles.

- **Extraction des Itemsets fréquents :**

Durant cette étape, on détermine les Itemsets fréquents ; c'est à dire les sous-ensembles d'articles qui apparaissent le plus fréquemment (par rapport à un *Support* minimal fixé par l'utilisateur) dans la base de données. C'est l'étape la plus coûteuse en terme de temps d'exécution car le nombre d'Itemsets fréquents dépend exponentiellement du nombre d'items manipulés (pour N items, on a 2^N -1 Itemsets potentiellement fréquents).

Depuis 1994, l'année où le premier algorithme d'extraction des règles d'association « Apriori » a été élaboré, les chercheurs travaillent afin de rendre cette étape moins coûteuse.

Une centaine d'algorithmes et de techniques ont vu le jour, certains ont été testés sur des bases de données expérimentales mais d'autres ont été appliqués dans différents domaines (Textmining, Webmining, en industrie, en biologie) à des bases de données réelles.

- **Génération des règles d'association :**

A partir de l'ensemble des Itemsets fréquents pour un seuil minimal de Support minsup, on génère les règles d'association vérifiant un seuil de confiance minimal. Le premier algorithme générateur de règles d'association Gen-règles [DUQ 99] a été élaboré par Agrawal en même temps qu'Apriori. D'autres algorithmes l'ont succédé comme AOP2 …etc.

Deux problèmes majeurs se posent au niveau de cette étape :

- Le premier est que pour un Itemset fréquent, la taille de l'ensemble des règles déduites est une fonction exponentielle de la taille de l'Itemset fréquent. En effet, si f est un Itemset fréquent de taille supérieure à 1, alors le nombre des règles qui y sont associées est $2^{|f|}$-2.
- Le second est la redondance des règles d'association. En effet, pour un ensemble F d'Itemsets fréquents, le nombre des règles d'association extraites est [$\sum_{f\in F)} 2^{|f|}$-2]. Ce nombre peut valoir plusieurs millions pour des bases de données normales. Mais plusieurs de ces règles représentent des redondances et n'ajoutent aucune information supplémentaire.

Certaines solutions sont proposées pour tenter de résoudre ces deux problèmes. Une solution consiste à utiliser des contraintes (Lift, Confiance Centrée) définies par l'utilisateur pour limiter le nombre d'ensemble de règles à considérer tout en lui fournissant un classement de l'intérêt des règles selon l'indice de pertinence. Une autre solution est la génération des règles informatives qui basé sur l'utilisation des représentations condensées (ensembles fermés fréquents) pour réduire le nombre de règles générées [Pas 00].

- **Visualisation et interprétation des règles d'association :**

Cette étape met entre les mains de l'utilisateur un ensemble de déductions fiables sous forme de règles qui peuvent l'aider à prendre des décisions.

Il faut que l'outil de visualisation prenne en compte la priorité des règles les unes par rapport aux autres, ainsi que les critères définis par l'utilisateur. De plus, il doit présenter les règles sous une forme claire et compréhensible.

4 Problématique : Extraction des Itemsets fréquents.

Le problème d'extraction de règles d'associations se résume en pratique dans l'étape d'extraction d'Itemsets fréquents et l'étape de génération des règles d'association à partir de l'ensemble des Itemsets fréquents.

L'étape d'extraction des Itemsets fréquents dans les bases de données est l'étape primordiale et la plus coûteuse dans le processus d'extraction des règles d'association. L'étape de génération des règles ne nécessite pas des accès à la base de données, et le temps d'exécution est beaucoup moins faible en comparaison avec le temps d'exécution de l'extraction des Itemsets fréquents. L'accent est mis donc dans les travaux de recherches sur l'étape d'extraction des Itemsets fréquents qui représente le principal cout de traitement.

Le problème d'extraction d'Itemsets fréquents s'appuis sur des mesures globaux (Support , fréquence…) et nécessite un accès à la totalité des données à traiter. Différents algorithmes d'extraction des Itemsets fréquents ont été proposés dans la littérature. Ces algorithmes parcours l'ensemble des Itemsets itérativement et par niveaux, à chaque niveau k, un ensemble d'Itemsets candidats de taille k est généré en joignant les Itemsets fréquents découverts durant l'itération précédente. Cet ensemble d'Itemsets candidats est élagué par la conjonction d'une métrique statistique (ex. le *Support*) et des heuristiques basées essentiellement sur les propriétés structurelles des Itemsets ; les *Support*s de ces Itemsets candidats sont calculés et les Itemsets non fréquents sont supprimés.

Cependant, cette étape est très couteuse du fait de la taille exponentielle de l'espace de recherche et du nombre élevé de balayages de la totalité de la base de données ; ce balayage est nécessaire pour évaluer les Supports des Itemsets et de calculer par la suite les confiances des règles générées.

Ces deux critères (le nombre de balayage et le nombre d'Itemsets générés) sont les deux facteurs d'efficacité ou d'inefficacité d'un algorithme d'extraction des Itemsets fréquents.

Le principe de base des algorithmes d'extraction des Itemsets fréquents est itératif et consiste à répéter les actions :

- Utiliser les (k-1)-Itemsets pour générer les k-Itemsets candidats ;
- Scanner les transactions pour calculer les *Support*s des candidats.

La phase de génération des candidats dans les algorithmes de recherche des Itemsets fréquents se décompose en deux sous-phases : la Construction de la liste des Itemsets candidats et l'Elagage de la liste des Itemsets candidats.

Les algorithmes (Apriori et ses dérivés) procèdent de manière itérative et se basent sur les propriétés suivantes afin de limiter le nombre de candidats considérés à chaque itération :

- **Propriété 1 : Tous les sous-ensembles d'un Itemset fréquent sont fréquents.**

Cette propriété est exploitée pour la phase de construction des Itemsets.

Le nombre d'Itemsets possible est 2^m -1 (*m* étant le nombre d'items présents dans la base), la **propriété 1** permet de limiter le nombre de candidats générés lors de la k^{ème} itération. Le principe de l'algorithme itératif permet de ne pas générer la totalité des Itemsets, on n'examine donc pas 2^m -1 Itemsets mais beaucoup moins.

Le nombre de 2-Itemsets possible est :

$$C_m^2 = \frac{m!}{(m-2)!\,2!} = \frac{m(m-1)}{2}$$

Pour chaque niveau k, on ne génère donc pas tous les k-Itemsets mais seulement un sous ensemble, basé sur le nombre de (k-1)-Itemsets fréquents identifiés au niveau précédent.

A l'itération 1 et pour *m* items de la base, on identifie *m* 1-Itemsets candidats. Le calcul des Supports de ces candidats via l'examen de l'ensemble des données permet d'identifier *m'* 1-Itemset fréquents (avec $m' \leq m$).

Or, on ne génère qu'un sous-ensemble des 2-Itemsets en se basant sur la propriété 1 : tous les sous-ensembles d'un Itemset fréquents sont fréquents.

Ainsi, on génère les candidats de taille 2 à partir des *m'* 1-Itemset fréquents, soit $\frac{m'(m'-1)}{2}$ 2-Itemsets candidats :

$$\frac{m'(m'-1)}{2} \leq \frac{m\ (m\ -1)}{2} \text{ puisque } m' \leq m.$$

De plus, on ne poursuit pas obligatoirement la génération jusqu'au Itemsets de taille m, on s'arrête à la génération des k-Itemsets (ou k < m).

Le critère d'arrêt de l'algorithme est lié à l'impossibilité de générer des candidats de taille k+1.

- **Propriété 2 : Tous les sur-ensembles d'un Itemset non fréquent sont non fréquents.**

Cette propriété est exploitée pour l'élagage de la liste des Itemsets candidats.

La **propriété 2** est utilisée pour l'élagage des candidats, elle permet la suppression d'un candidat de taille k lorsqu'au moins un de ses sous-ensembles de taille k-1 ne fait pas partie des Itemsets fréquents découverts précédemment.

L'algorithme Apriori introduit par Agrawal [AGR 94], est l'algorithme de référence et le premier algorithme d'extraction des règles d'association dans les bases de données transactionnelles.

L'idée générale de cet algorithme est que les attributs sont triés dans un ordre lexicographique. En effet, au début les Supports des 1-Itemsets sont calculés en effectuant une première passe sur la base de données. Les 1-Itemsets, dont le Support n'a pas atteint le seuil *Minsup*, sont écartés, c'est-à-dire, ils sont considérés comme non fréquents.

```
L1 = { 1-Itemsets fréquents};
Pour (k= 2; Lk-1≠Ø; k++) faire
    Ck= apriori-gen(Lk-1);          // la phase de génération-élagage
    pour chaque transaction t ∈ D faire
        Ct= sous-ensemble (Ck,t);    //les candidats contenus dans la transaction t
        pour chaque candidates c ∈ Ct faire
                    c.freq++ ;
    fin;
    Lk={c∈ Ck |c.sup ≥ minsup}
Fin
Résultats= ∪k Lk
```

Fig. 1.5 L'algorithme Apriori

Par la suite, un nouvel ensemble des 2-Itemsets, appelé ensemble des 2-Itemsets candidats, est généré. Après une autre passe à travers la base, les Supports des 2-Itemsets sont calculés. Les 2-Itemsets non fréquents sont écartés et le processus décrit précédemment est relancé, jusqu'au moment où l'on ne peut plus générer d'Itemsets fréquents.

Nous allons présenter un exemple de l'algorithme Apriori pour illustrer ces propriétés. Soit le contexte d'extraction β, on cherche à extraire les Itemsets fréquents pour un *Support* minimal *Minsup* =2/6 :

DATABASE β

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

Au début, tous les items de la base de données sont considérés comme des 1-Itemset candidats et donc leurs Supports sont calculés en effectuant une première passe sur la base de données. Les Itemsets, dont le Support n'a pas atteint le seuil *Minsup*, sont écartés, c'est-à-dire, ils sont considérés comme non fréquents :

Candidats C_1
A
B
C
D
E

Balayage De β →

C_1	*Support*
A	3/6
B	5/6
C	5/6
D	1/6
E	5/6

Suppression des Items non fréquents →

Frequents F_1	*Support*
A	3/6
B	5/6
C	5/6
D	1/6
E	5/6

Par la suite, un nouvel ensemble des 2-Itemsets, appelé ensemble des Itemsets candidats C_2, est généré par auto-jointure (phase de génération-élagage) des Itemsets fréquents trouvés à l'étape précédente.

Cet ensemble est construit par la fusion des deux (k-1)-Itemsets **p** et ***q*** qui partagent leur (k-2)-premiers items:

Insert into C_k select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$
From L_{k-1} p, L_{k-1} q
Where p.item$_1$=q.item$_1$, …, p.item$_{k-2}$=q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$

En suite, on supprime de cet ensemble C_k tout Itemset *c* pour lequel au moins un sous-ensemble de longueur k-1 de *c* n'appartient pas à la liste des Itemsets fréquent de l'itération précédente L_{k-1} :

Pour chaque Itemset c dans C_k Faire
Pour chaque (k-1)-sous-ensemble s de c Faire
Si (s n'est pas dans L_{k-1}) Alors supprimer c de C_k;

Après une autre passe à travers la base, les *Support*s des 2-Itemsets sont calculés. Les 2-Itemsets non fréquents sont écartés :

Candidats C_2
AB
AC
AE
BC
BE
CE

Balayage De β →

C_2	*Support*
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

Suppression des Items non fréquents →

Frequents F_2	*Support*
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

Ce processus décrit est relancé, jusqu'au moment où l'on ne peut plus générer d'Itemsets fréquents.

Candidats C_3
ABC
ABE
ACE
BCE

Balayage De β →

C_3	*Support*
ABC	**2/6**
ABE	2/6
ACE	2/6
BCE	4/6

Suppression des Items non fréquents →

Frequents F_3	*Support*
ABC	**2/6**
ABE	2/6
ACE	2/6
BCE	4/6

Candidats C_4
ABCE

Balayage De β →

C_4	*Support*
ABCE	2/6

Suppression des Items non fréquents →

Frequents F_4	*Support*
ABCE	2/6

Cet algorithme souffre de la gestion du nombre de candidats qu'elle pourrait générer, surtout pour des contextes fortement corrélés et/ou des valeurs de *Support* relativement faibles. Ce constat est renforcé par des accès disque répétitifs pour le calcul du *Support* des candidats.

Toutefois, cet algorithme doit déterminer le *Support* de tous les Itemsets fréquents et même de certains Itemsets non fréquents. Un certain nombre de modifications ont été proposées afin d'améliorer divers aspects, essentiellement sur la réduction du coût des entrées/sorties à la base de données et la minimisation du coût de l'étape de calcul du *Support*.

Pour plus de détails sur ces algorithmes, on peut se référer essentiellement à AprioriTID [HIP 00], Partition [SAV 95], Sampling [SAV 95], DIC [BRI 97], ECLAT [ZAK 97], FP-ROWTH [HAN00].

5 Approche

Notre approche consiste en la modification de la phase de la génération-élagage des Itemsets de l'algorithme Apriori et ses dérivées en introduisant une nouvelle méthode pour la génération des Itemsets candidats qui se base sur la notion de ***Support***$^+$.

Nous avons remarqué que les algorithmes qui se base sur l'algorithme Apriori, lors de la phase de construction des Itemsets candidats de taille k utilisent les deux Itemsets fréquent de taille k-1 ayant (k-2) items préfixe en commun; et examine en suite pour chaque candidat construit si au moins un de ses sous-ensembles de taille k-1 n'appartient pas à l'ensemble des Itemset fréquents de taille k-1.

Selon la propriété 1, un k-Itemsets Y ne peut être fréquent si au moins un des ses sous-ensembles ne l'est pas. Nous avons donc constaté que pour un K-Itemset Y, il suffit d'identifier le (k-1)-sous-ensemble de Y qui a la plus petite valeur de *Support* pour en déduire si Y est fréquent ou pas.

Cette valeur représente la valeur minimale de *Support* qui peut prendre le K-Itemset Y, nous l'avons appelée le ***Support***$^+$ du k-Itemset Y.

Par conséquent, nous avons introduit une nouvelle technique pour le problème d'extraction d'Itemsets fréquents en se basant sur le ***Support***$^+$. Nous avons utilisée cette valeur de ***Support***$^+$ d'une part, dans la phase d'élagage des Itemsets candidats pour déterminer si un k-Itemset ne peut ~~pas~~ être fréquent.

D'autre part, nous avons exploité ce ***Support***$^+$ pour l'extraction des Itemsets fréquent en se basant non pas sur la métrique de Support qui nécessite l'accès à la base de données mais sur le ***Support***$^+$ qui donne une valeur approximative du Support réel et qui est déduit des propriétés des Itemsets et sans l'accès à la base de données.

Or, une phase de validation est nécessaire pour calculer les supports exacts des Itemsets fréquents trouvés et pour calculer les confiances des règles d'association à générer dans la prochaine phase. Deux versions d'algorithme ont été implémentées et testées, à savoir : une version avec validation à posteriori et une version avec validation par itération.

Nous présentons dans cette section le principe général de notre approche et les deux versions d'algorithmes séquentiels.

5.1 Notion de *Support*$^+$

Nous avons introduit une technique pour l'estimation des *Support*s des Itemsets. Cette technique est déduite des propriétés des Itemsets. Nous allons utiliser cette technique dans la phase de génération-élagage des candidats.

Nous introduisons la propriété suivante :

- **Propriété 3 : Etant donné un Itemset Y, le *Support* d'un sous-ensemble *x* de Y est supérieur ou égal au *Support* de Y.**

Justification :

Soit *D* une base de données de *n* transactions et prenons comme exemple de *D* le 3-Itemset ***xyz***. On note par T_{xyz} l'ensemble des transactions de *D* qui contiennent le triplet d'items (*x*,*y*,*z*) et $Card(T_{xyz})$ la cardinalité de l'ensemble T_{xyz}.

Nous avons donc le Support du 3-Itemset ***xyz*** : $Support(\mathbf{xyz}) = Card(T_{xyz})/n$.

Le 3-Itemset ***xyz*** possède trois sous-ensembles de taille 2 (***xy***, ***xz***, ***yz)***; soit T_{xy}, T_{xz}, T_{yz} les ensembles de transactions de *D* contenant ces sous-ensembles avec les cardinalités respectives $Card(T_{xy})$, $Card(T_{xz})$, $Card(T_{yz})$ et les *Support*s suivants :

$Support(\mathbf{xy}) = Card(T_{xy})/n$, $Support(\mathbf{xz}) = Card(T_{xz})/n$, $Support(\mathbf{yz}) = Card(T_{yz})/n$.

L'ensemble T_{xyz} est un sous ensemble de l'ensemble T_{xy} : toutes les instances contenant le triplet d'items (*x*,*y*,*z*) contiennent obligatoirement le couple d'items (*x*,*y*).

Par contre, toutes les transactions contenant le couple d'itemes (*x*,*y*) ne contiennent pas forcémenent l'item ***z***, et donc ne contient pas forcément le triplet d'items (*x*,*y*,*z*).

Donc $T_{xyz} \subseteq T_{xy}$, d'où : $Card(T_{xyz}) \leq Card(T_{xy})$

et par association (*n* étant positif) : $Card(T_{xyz})/n \leq Card(T_{xy})/n$

et donc $Support(\mathbf{xyz}) \leq Support(\mathbf{xy})$.

Par conséquent, on a bien le *Support* du sous-*ensemble* ***xy*** supérieur ou égal au *Support* du sur-ensemble ***xyz*** : $Support(\mathbf{xyz}) \leq Support(\mathbf{xy})$.

De même pour les sous-ensembles ***xz*** et ***yz***, les *Support*s de ***xz*** et ***yz*** sont supérieures ou égaux au *Support* du sur-ensemble ***xyz***.

Nous avons etendu cette propriété pour définir la notion de ***Support***$^+$:

Soit le 3-Itemset ***xyz*** et ***xy***, ***xz***, ***yz*** les 2-Itemsets sous-ensemble de ***xyz***. Nous avons selon la propriété 3 les relations suivantes :

Supp (***xyz***) ≤ Supp (***xy***) : ***xy*** est un sous-ensemble de ***xyz***. (i)

Supp (***xyz***) ≤ Supp (***xz***) : ***xz*** est un sous-ensemble de ***xyz***. (ii)

Supp (***xyz***) ≤ Supp (***yz***) : ***yz*** est un sous-ensemble de ***xyz***. (iii)

De (i), (ii) et (iii) on déduit la relation suivante :

$$\text{Supp (xyz)} \leq \min\{\text{Supp (xy)}, \text{Supp (xz)}, \text{Supp (yz)}\} \qquad (a.1)$$

Et donc nous introduisant la proposition suivante :

- **Proposition 1 : Etant donné un k-Itemset Y, le *Support* de Y est inférieur ou égale au minimum des *Support*s des (k-1)-Itemsets sous ensembles de Y.**

Ce minimum représente la valeur minimale de *Support* que peut prendre le sur-ensemble Y. on propose d'appeler cette valeur le ***Support***$^+$.

Donc, le *Support*$^+$ du 3-Itemset ***xyz*** est défini comme suit:

$$Support^+(\boldsymbol{xyz}) = \min\{\text{Supp } (\boldsymbol{xy}), \text{Supp } (\boldsymbol{xz}), \text{Supp } (\boldsymbol{yz})\}. \qquad \textbf{(a.2)}$$

et ainsi, on introduit la proposition 2 :

- **Proposition 2 : Le *Support*$^+$ d'un k-Itemset Y est égal au minimum des *Support*s des (k-1)-Itemsets sous ensembles de Y.**

Le ***Support***$^+$ d'un Itemset Y est défini comme suit :

$$Support^+(Y) = \min_{|x|=|y|-1}\{Support(x)\} \qquad (b)$$

5.2 Élagage des Itemsets candidats par le *Support*$^+$:

Le ***Support***$^+$ d'un k-Itemset Y nous donne une estimation de la valeur du *Support* de l'Itemset Y. Sa valeur est déduite à partir du minimum de ses k-1 sous-ensembles trouvés dans l'itération précédente et donc aucun accès à la base de données n'est nécessaire pour le calcul de cette valeur.

Cette notion de ***Support***$^+$ est utilisée pour l'extraction des Itemsets fréquents particulièrement pour la phase de génération-élagage des Itemsets candidats.

Nous reprenons les relations (a.1) et de (a.2) précédentes pour en déduire la relation (a.3) suivante :

$$Support(\boldsymbol{xyz}) \leq \boldsymbol{Support}^+(\boldsymbol{xyz}) \qquad (a.3).$$

En analysant la relation (a.3), on voit bien que si le ***Support***$^+$ du 3-Itemset ***xyz*** est inférieur au seuil minimum de *Support Minsup* alors on peut déduire que le *Support* de ***xyz*** est inférieur à *Minsup* et par conséquence, le 3-Itemset ***xyz*** n'est pas fréquent :

Si $Support^+(\boldsymbol{xyz}) \leq Minsup$ **Alors** $Support(\boldsymbol{xyz}) \leq Minsup$.

Ainsi on introduit la propriété suivante :

❖ **Propriété 4 : Un Itemset Y n'est pas fréquent si son *Support*$^+$ est inférieur au seuil de *Support* minimale *Minsup*.**

Justification :

Considérons par exemple le 3-Itemset ***xyz*** et ses trois sous-ensembles de taille 2 (***xy***, ***xz*** , ***yz)***.

Si le 3-Itemset ***xyz*** n'est pas *fréquent* alors obligatoirement le *Support* (***xyz***) ≤ *Minsup*.

Donc au moins un des *Support*s de ses sous-ensembles (*Support* (***xy***), *Support* (***xz***), *Support* (***yz***) est inférieur à *Minsup.*

Or, le **Support**$^+$ (**xyz**)=min { Support (**xy**), Support (**xz**), Support (**yz**)}, donc il est evident que le **Support**$^+$(**xyz**) ≤ Minsup .

Cette propriété 4 est exploitée dans la génération-élagage des Itemsets candidats. en examinant la valeur de ***Support***$^+$ de l'Itemset Y, et si cette valeur est inférieur au seuil minimal de Support *Minsup,* on déduit directement que l'Itemset Y n'est pas fréquent et donc cet Itemset n'est pas généré comme candidat:

$Support^+(Y) \leq Minsup \Rightarrow Support(Y) \leq Minsup.$ (c).

5.3 Le Calcul de *Support*$^+$:

L'utilité du ***Support***$^+$ réside dans le fait que le calcul de sa valeur ne nécessite pas l'accès aux sources de données mais simplement déduit à partir de l'itération précédente et que ce *Support*$^+$ est utilisé pour l'élagage des candidats.

Le calcul de la valeur de ***Support***$^+$ est donné par la formule suivante :

$$Support^+(Y) = \min_{|x|=|y|-1} \{support(x)\} \quad \text{(b)}$$

Cependant, on mentionne ici que le calcul de ***Support***$^+$ des Itemsets candidats n'est intéressant que pour les Itemsets de taille supérieur ou égale à 3 (les 3-Itemsets ou plus).

-Les 1-Itemset n'ont pas de sous-ensembles.

-Les 2-Itemsets ont exactement deux sous-ensembles de taille 1 ; chaque 2-Itemset candidat est généré a partir de deux 1-Itemsets fréquents et donc il est inutile d'utiliser le ***Support***$^+$ pour

l'élagage puisque forcément le minimum des Supports de ses sous-ensembles est supérieur ou égal au Minsup.

Par conséquent, nous avons introduit la structure de données **CountMap** pour faciliter le calcul des 1-Itemsets et 2-Itemsets fréquents que nous expliquons plus loin dans ce chapitre.

La stratégie de l'algorithme Apriori et ses dérivés est simple. On génère les Itemsets potentiellement fréquent d'une taille k à partir des k-1 Itemsets fréquents trouvés dans l'itération k-1. En suite on parcourt la base de données pour calculer les *Support*s de ces Itemsets. A la fin, on ne garde que les Itemsets qui satisfont le seuil minimal de *Support Minsup*.

Prenons l'exemple précédent de l'algorithme Apriori avec la base de données **β** et un *Minsup*=2/6.

L'exécution de l'itération 1 et 2 nous permet d'identifier les 1-Itemset et les 2-Itemsets fréquents :

DATABASE **β**

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

Frequents F_1	*Support*
A	3/6
B	5/6
C	5/6
E	5/6

Frequents F_2	*Support*
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

A partir de l'itération 3, on peut calculer la valeur de ***Support***$^+$ des k-Itemsets. Les colonnes en pointillés représentent les valeurs de *Support*$^+$ des Itemsets candidats estimées selon la formule (b):

C_3	*Support*$^+$
ABC	min(2/6,3/6,4/6)=**2/6**
ABE	min(2/6,2/6,5/6)=**2/6**
ACE	min(3/6,2/6,4/6)=**2/6**
BCE	min(4/6,5/6,4/6)=**4/6**

Balayage De **β** →

C_3	*Support*
ABC	2/6
ABE	2/6
ACE	2/6
BCE	4/6

Suppression des Items non fréquents →

F_3	*Support*
ABC	**2/6**
ABE	**2/6**
ACE	**2/6**
BCE	**4/6**

De même pour les 4-Itemsets :

C_4	*Support*$^+$
ABCE	min(2/6,2/6,2/6,4/6)=**2/6**

Balayage De **β** →

C_4	*Support*
ABCE	2/6

Suppression des Items non fréquents →

F_4	*Support*
ABCE	**2/6**

On remarque que les valeurs de ***Support***$^+$ estimées des Itemsets sont très proches des valeurs exactes. En plus, le calcul de ***Support***$^+$ ne nécessite aucun accès aux sources de données mais des simples déductions à partir des résultats trouvés dans l'itération précédente de l'algorithme.

Donc, on propose d'utiliser ces *Support*s approximatives (***Support***$^+$) à la place des Supports exactes pour la recherche des Itemsets fréquents : on calcul les ***Support***$^+$ des Itemsets candidats, en éliminant par la suite ceux qui ne satisfont pas le seuil minimum de Support *Minsup*. On répète itérativement ce processus jusqu'à ce qu'aucun Itemset ne peut être généré.

Le calcul de ***Support***$^+$ présenté par la formule (b) repose sur les résultats de l'itération précédente (les *Support* exactes de l'itération k-1). Ainsi il faut faire un balayage de la base de données pour calculer les Supports des k-1 sous-ensembles.

Par conséquence, nous avons proposé une autre méthode pour calculer la valeur de ***Support***$^+$ on utilisant une fonction récursive qui nous donne des valeurs approximatives de *Support*+.

Soit le 3-Itemset ***xyz*** , avec ses 2-Itemsets sous-ensemble ***xy***, ***xz*** et ***yz***;

Partons de la formule (b) : $Support^+(Y) = \min_{|x|=|y|-1} \{support(x)\}$

Nous avons : ***Support***$^+$(***xyz***)= min{ Supp (***xy***), Supp (***xz***), Supp (***yz***)}.

On cherche à remplacer dans cette formule le *Support*(***xy***) (de même pour ***xz*** et ***yz)*** par le ***Support***$^+$ (***xy***) afin d'éviter toute dépendance des résultats de l'itération précédente.

On rappelle que selon la relation (a.3) :

Support(**xy**) ≤ **Support**$^+$ (**xy**)

Support(**xz**) ≤ **Support**$^+$ (**xz**)

Support(**yz**) ≤ **Support**$^+$ (**yz**).

Dans ce cas, deux cas de figure se présentent :

- **Support(xy) = Support$^+$(xy) (de même pour xz et yz) :**

  Dans ce cas l'équation ***Support***$^+$(***xyz***)= min{ *Support*(*xy*), *Support*(*xz*), *Support*(*yz*)} et l'équation ***Support***$^+$(***xyz***) = min{ ***Support***$^+$(*xy*), ***Support***$^+$(*xz*), ***Support***$^+$(*yz*)} sont touts les deux équivalentes.
  Et donc la formule suivante est équivalente à la formule (b) :

$$Support^+(Y) = \min_{|x|=|y|-1} \{Support^+(x)\}$$

- **Support(xy) < Support$^+$(xy) (de même pour xz et yz):**

  Dans ce cas le ***Support***$^+$(***xyz***) < min{ ***Support***$^+$(***xy***), ***Support***$^+$(***xz***), ***Support***$^+$(***yz***)}

Cependant, on se rappelle que (dans l'exemple précédent) la valeur de ***Support***$^+$ est très proche de la valeur du Support exacte des Itemsets et donc on a la relation suivante :

$$Support^+(Y) \cong \min_{|x|=|y|-1} \{Support^+(x)\}$$

On propose à remplacer la formule (b) par une autre formule récursive qui donne une approximation de la valeur de ***Support***$^+$ et sans se référer au Support exact des Itemsets précédents.

Nous avons donc introduit la proposition suivante:

- **Proposition 3 :Le *Support*$^+$ d'un k-Itemset Y est approximativement égale au minimum des *Support*s$^+$ des (k-1)-Itemsets sous ensemble de Y.**

Ainsi, on propose de calculer la valeur de ***Support***$^+$ par la formule (d) suivante :

$$Support^+(Y) = min_{|x|=|y|-1}\ \{Support^+(x)\} \qquad (d)$$

5.4 Approche pour l'extraction des Itemsets fréquents :

Les algorithmes de recherche des Itemsets fréquents procèdent itérativement par niveau, à chaque itération une liste d'Itemsets candidats de taille k est construite et élaguée par la suite. Nous avons introduit une nouvelle méthode pour l'extraction des Itemsets fréquents basée sur la notion de ***Support***$^+$. Nous avons introduit une optimisation dans la phase de génération-élagage des candidats en exploitant la notion de ***Support***$^+$.

Le *Support*$^+$ donne des valeurs approximatives des Supports réel des Itemsets. en se basant sur la proposition 4 : **Un Itemset Y n'est pas fréquent si son Suppot$^+$ est inférieur au seuil de suppot minimal *Minsup*,** on propose donc de considérer un Itemset (approximativement) fréquent si son *Support*$^+$ est supérieur au seuil de Support minimal *Minsup*.

La méthode que nous avons introduite pour le calcul des Itemsets fréquents est basée sur la notion de *Support*$^+$. Le *Support*$^+$ nous donne une valeur approximative du *Support*. Or, comme nous avons expliqué auparavant, le calcul de ***Support***$^+$ des Itemsets candidats n'est intéressant qu'aux Itemsets de taille supérieure ou égale à 3 (les 3-Itemsets ou plus) :

- Les 1-Itemsets n'ont pas de sous-ensembles.
- Les 2-Itemsets ont exactement deux sous ensembles de taille 1 et que ces deux sous-ensembles sont les seuls utilisés pour construire les 2-Itemsets. On rappel que l'idée d'utilisation du ***Support***$^+$ est de détecter la présence d'autres sous-ensembles non fréquents en dehors des deux sous-ensembles utilisés pour construire l'Itemset en question.

Par conséquence, nous avons introduit la structure de données CountMap pour le calcul des 1-Itemset et des 2-Itemsets fréquents. De plus, la structure de données CoutMap est utilisée pour optimiser le calcul de ***Support***$^+$ des Itemsets candidats sans y accéder à la base de données.

5.4.1 Le calcul des k-Itemsets (k<3) :

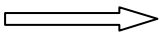
La structure CountMap est une matrice à deux dimensions de taille (*m*×*m*), *m* représente le nombre d'items de la base de données. La matrice CountMap représente une projection de la base de données. Chaque cellule (i,j) de CountMap correspond à la fréquence de l'Itemset constituée des éléments '*yi yj*'.

La figure suivante illustre un exemple de CountMap pour un alphabet *I*={A,B,C,D ,E} (*m*=5 éléments) et pour la base de données présentée dans la table **β**.

Les cellules représentent les fréquences des 1-Itemsets (les cellules de la diagonale) et des 2-Itemsets (les cellules supérieurs à la diagonale de la matrice):

DATABASE $\boldsymbol{\beta}$

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE



COUNTMAP

	A	**B**	**C**	**D**	**E**
A	3	2	3	1	2
B	-	5	4	0	5
C	-	-	5	1	4
D	-	-	-	1	0
E	-	-	-	-	5

La structure CountMap est construite en parcourant la base de données $\boldsymbol{\beta}$. Pour chaque transaction, on incrémente la fréquence du 1-Itemset *yi* et les différents 2-Itemset *yi yj* contenus dans la transaction.

À la fin du parcours de toute la base de données, les Supports exacts des 1-Itemsets et des 2-Itemsets sont immédiatement identifiés :

$$Support\ (y_i) = CountMap(i,i)/n, \quad |y_i| = 1$$
$$Support\ (y_i y_j) = CountMap(i,j)/n, \quad |y_i y_j| = 2$$

Avec $n = |\boldsymbol{\beta}|$

Par exemple :

$Support(A) = CountMap(1,1)/6 = 3/6,$ $Support(E) = CountMap(5,5)/6 = 5/6$
$Support(AB) = CountMap(1,2)/6 = 2/6$ $Support(AE) = CountMap(1,5)/6 = 2/6$

Dans notre approche un parcours entier de la base de données est effectué pour construire la structure CountMap. Ensuite, on calcule les Supports exacte des 1-Itemsets par un simple accès à la structure CountMap et on élimine ceux qui ne satisfont pas le seuil de Support minimal.

On génère en suite les 2-Itemset candidats à partir de ces 1-Itemsets fréquents et on récupère leurs Supports exacts à partir de CountMap. On ne garde à la fin que les 2-Itemsets fréquents.

5.4.2 Le calcul des k-Itemsets fréquent (k≥3) :

En suivant le principe itératif d'algorithme d'extraction des Itemsets fréquents, on génère les Itemsets candidats d'un niveau k et on calcule leurs valeurs de ***Support***$^+$ et on considère qu'un k-Itemsets est (approximativement) fréquents si son ***Support***$^+$ est supérieur au seuil de Support minimal *Minsup* :

Insert into C_k **Select** $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
From L_{k-1} p, L_{k-1} q
Where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
and $Support^+(p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}) \geq Minsup$

On rappelle ici que la génération des Itemsetselon le *Support*$^+$ commence à partir des Itemset de taille 3, les Itemsets de taille 1 et 2 sont extraits à partir de la structure CountMap.

On peut utiliser la formule (b) pour calculer les ***Support***$^+$ des 3-Itemsets puisque les 2-Itemset sous-ensembles sont déjà identifies via la structure CounMap:

$$Support^+(\mathrm{Y}) = \min_{|\mathrm{x}|=|\mathrm{y}|-1} Support \quad (\mathrm{x})$$

Ainsi, le ***Support***$^+$ du 3-Itemset xyz est définit par :

$$Support^+(\mathrm{xyz}) = \min\{ Support \quad (\mathrm{xy}), Support \quad (\mathrm{xz}), Support \quad (\mathrm{yz})\}$$

$$Support^+(\mathrm{xyz}) = \min\{\mathrm{CountMap}(\mathrm{i_x}, \mathrm{i_y})/\mathrm{n}, \mathrm{CountMap}(\mathrm{i_x}, \mathrm{i_z})/\mathrm{n}, \mathrm{CountMap}(\mathrm{i_y}, \mathrm{i_z})/\mathrm{n}\}$$

Avec i_x , i_y et i_z l'ordre lexicographique de l'item x, y et z, et *n* le nombre de transactions de la base.

Pour les k-Itemset de taille supérieur à 3 (k>3), le calcul des ***Support***$^+$ est effectuer via la formule (d) afin d'éviter les accès aux sources de données pour la récupération des Supports exacts des (k-1)-Itemset sous ensembles :

$$Support^+(Y) = min_{|x|=|y|-1} \{Support^+(x)\} \qquad (d)$$

Donc, le calcul du ***Support***$^+$ des k-Itemsets est effectué de façon récursive comme suit :

$$Support^+(\mathrm{Y}) = \begin{cases} \min_{|\mathrm{x}|=|\mathrm{y}|-1} Support \quad (\mathrm{x}), & |\mathrm{Y}| = 3 \\ \min_{|\mathrm{x}|=|\mathrm{y}|-1} Support^+ \ (\mathrm{x}), & |\mathrm{Y}| > 3 \end{cases}$$

5.4.3 La phase de validation :

Une fois que le calcul des *Support*$^+$ des Itemsets du niveau k est terminé, on élimine les Itemsets ayant des *Support*$^+$ inférieur à *Minsup* (élagage par le *Support*$^+$). Les Itemsets non éliminés sont utilisés pour générer les Itemset du niveau k+1.

Cependant, l'utilisation de ***Support***$^+$ pour la recherche des Itemsets fréquents génère certains Itemsets non fréquents comme étant des Itemsets fréquents car le ***Support***$^+$ donne une approximation du Support réel de l'Itemset. Une phase de validation est nécessaire pour calculer les Supports exacts des Itemsets fréquents trouvés et pour éliminer les Itemsets non fréquents qui surviennent de la phase d'élagage.

Deux cas de figure se présentent lorsque le ***Support***$^+$ est supérieur au *Minsup* :

- **Cas 1 : *Minsup* ≤ *Support*(Y) ≤ *Support*$^+$ (Y)**

  → L'Itemset Y est fréquent et donc il faut calculer son Support exact qui va servir par la suite pour la génération des règles d'association.

- **Cas 2 : *Support* (Y) ≤ *Minsup* ≤ *Support*$^+$ (Y)**

  → L'Itemset Y n'est pas fréquent, mais il est généré et il faut l'éliminer de l'ensemble des Itemsets fréquents.

Donc un parcours de la totalité de la base de données doit être effectué pour raffiner l'ensemble des k-Itemsets fréquents et calculer leurs Supports exacts.

Deux alternatives se présentent pour la phase de validation :

- **A la fin de génération de tous les k-Itemsets :**

  On génère itérativement et par niveaux les k-Itemset de taille supérieur ou égale à 3 ($k \geq 3$). A chaque itération k, on calcule les *Supports*$^+$ des k-Itemsets. Une fois terminé, on élimine ceux ayant des *Support*$^+$ inférieur à *Minsup* (propriété 4). Les Itemsets non éliminés sont utilisés pour générer les Itemsets du niveau k+1 et ce processus est répété jusqu'à ce qu'aucun Itemset ne peut être généré.

  A la fin de calcul de tous les k-Itemsets fréquents par l'utilisation de *Support*$^+$, une phase de validation est effectuée. Donc, un seul parcours de la base de données est nécessaire pour raffiner l'ensemble des Itemsets fréquents calculés.

  Dans ce cas de figure, le nombre d'accès aux sources de données est réduit. Cela implique aussi qu'on a besoin de moins de communications et de synchronisation lorsque la base de données est distribuée à travers plusieurs sites.

Cependant, cette stratégie peut générer dans certaines situations un nombre important d'Itemsets parasites : des Itemsets non fréquents qui survivent de l'élagage par l'utilisation de *Support*$^+$.

- **A chaque itération :**

  Dans la deuxième alternative, la phase de validation est effectuée à chaque itération de la phase de calcul des Itemsets fréquents par l'utilisation des $Support^+$. Ici, l'ensemble des Itemsets est raffiné à chaque itération et donc l'ensemble des Itemsets parasites est très réduit.

  Cela implique un accès aux sources de données à chaque itération de l'algorithme et donc des communication/synchronisations entre les différents sites dans le cas ou la base de données est distribuée entre ces sites.

En analysant ce cas de figure, on revient à une version semblable à l'algorithme Apriori. Cependant, les expérimentations et les tests effectués ont montré que le nombre de candidats générés par notre approche est plus réduit que le nombre de candidats générés par la version classique d'Apriori.

Une hybridation de ces deux approches semble intéressante en fixant un seuil $S_{Candidats}$ sur le nombre maximal de candidats considérés par l'algorithme. On génère itérativement les Itemsets fréquents par l'utilisation de $Support^+$ sans aucune validation. Une fois que le seuil de nombre de candidats $S_{Candidats}$ est atteint, on bascule vers le deuxième mode en lançant la phase de validation pour raffiner les résultats. Ensuite, On revient ensuite au premier mode et ainsi de suite. On signale ici que le seuil $S_{Candidats}$ doit être soigneusement choisi. Cette version est étudiée dans le chapitre 3.

A la base des deux alternatives présentées, deux versions d'algorithmes d'extraction des Itemsets fréquents ont été proposées et testées, à savoir : une version avec validation à posteriori et une version avec validation par itération.

5.5 Algorithme proposé (AprioriCM):

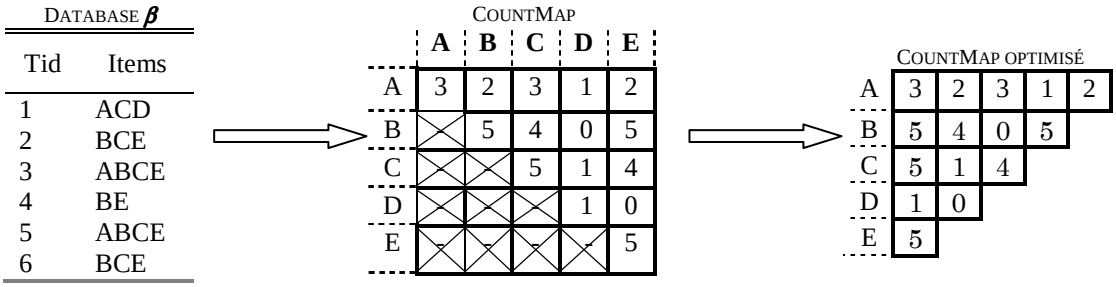
Dans cette section nous présentons le principe général de l'algorithme générique de notre approche : AprioriCM (Apriori CountMap). Nous avons modifié la version de base de l'algorithme Apriori en exploitant la notion de $Support^+$ qu'on a introduit et en implémentant la structure de données CountMap présentée auparavant.

Deux versions d'algorithme AprioriCM d'extraction des Itemsets fréquents sont proposés et testés et en se basant sur la manière où la phase de validation est effectuée. Dans cette section, nous allons présenter une comparaison entre la version originale d'Apriori et notre Algorithme AprioriCM (les deux versions) dans un contexte séquentiel afin d'illustrer le schéma général de notre approche.

5.5.1 Implémentation de la structure CountMap :

La structure CountMap est très sollicitée dans la phase d'élagage des Itemsets candidats. Sa taille dépend de la taille de l'alphabet (le nombre d'items) utilisée: pour un alphabet I de m éléments, sa taille est de l'ordre de ($m \times m$).

Au niveau de l'implémentation de la structure CountMap et vue que les Itemsets sont ordonnés par ordre lexicographique, nous avons éliminé les éléments redondants (symétriques) dans la matrice CountMap et donc sa taille est réduite à ½(m^2+m) comme illustré dans l'exemple suivant :



Ainsi l'accès à la structure CountMap sera modifié et pour un Itemsets constitué des éléments '*yi yj*', l'accès sera comme suit :

$$Support\ (y_i) \quad = CountMap(i,1)/n \qquad ,|y_i| \quad = 1$$
$$Support\ (y_i y_j) = CountMap(i, j-(i-1))/n \quad ,|y_i y_j| = 2$$

Par exemple :

$Support(E) = CountMap(5,1)/6 = 5/6$
$Support(BE) = CountMap\big(2,5-(2-1)\big)/6 = CountMap(2,4)/6 = 5/6$

5.5.2 AprioriCM: Version avec validation par itération

L'algorithme AprioriCM est semblable à l'algorithme Apriori avec une modification de la phase de génération-élagage des candidats.

```
1.  CM= Gen-CountMap(β ,I) ;
2.  L1 =GetFrequentItemsets(CM,1) ;      // calcul des 1-Itemsets fréquents
3.  L2=GetFrequentItemsets(CM,2) ;       // calcul des 2-Itemsets fréquents
4.  Pour (k= 3; Lk-1≠Ø; k++) faire
5.  L+k = CM-gen(Lk-1);                  // la phase de génération-élagage a base de support+
6.   pour chaque transaction t ∈ β faire
7.    Ct= sous-ensemble (L+k ,t);        //les candidats contenus dans la transaction t
8.     pour chaque candidates c ∈ Ct faire
9.                c.freq++;
10.    fin;
11.   fin;
12.  Lk={c∈ Ck |c.sup ≥ minsup}
13. Fin
14. Résultats= ∪k Lk
```

Fig 1.6 Algorithme AprioriCM

L'algorithme est basé sur la notion de *Support*$^+$ et il est constitué des étapes suivantes :

- **Etape 1 : Construction de la structure CountMap**

À cette étape, un parcours de la base de données est effectué pour construire la structure CountMap (*Gen-CountMap*). La structure CountMap est une projection de la base de données introduite pour le calcul des 1-Itemset et des 2-Itemsets fréquents.

Chaque cellule (i,j) de CountMap correspond à la fréquence de l'Itemset constituée des éléments 'yi yj'. Pour chaque transaction parcouru, on incrémente la fréquence du 1-Itemset 'yi' et les différents 2-Itemset 'yi yj' contenus dans cette transaction.

A la fin de ce parcours les *Support*s des 1-Itemsets et des 2-Itemsets sont calculés (*GetFrequentItemsets*) et ils sont immédiatement disponibles par un simple accès à CountMap et sans aucun autre parcours.

On remarque que le calcul des 1-Itemsets fréquents et des 2-Itemsets fréquents nécessite qu'un seul parcours de la base de données au lieu de deux parcours pour son homologue Apriori.

- **Etape 2 : Extraction des k-Itemsets fréquents**

On génère itérativement les k-Itemsets fréquents. On rappelle que le calcul des k-Itemsets commence à partir de l'itération k=3, les 1-Itemset et les 2-Itemsets sont déjà calculés dans l'étape précédente.

À chaque itération k (k≥3), deux sous étapes sont exécutées:

- **Etape 2.1 : génération des k-Itemsets candidats**

On construit les k-Itemsets candidats par auto-jointure des k-1-Itemsets générés dans l'itération précédente (*CM-gen*). L'ensemble des k-Itemsets candidat est élagué selon la formule (c) (élagage à base de *Support*$^+$) : On calcule pour chaque candidat le *Support*$^+$, si ce *Support*$^+$ est inférieur à *Minsup* alors ce candidat est écarté.

Insert into Ck Select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$
From L_{k-1} p, L_{k-1} q
Where p.item$_1$=q.item$_1$, …, p.item$_{k-2}$=q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$
and *Support*$^+$(p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$) ≥ *Minsup*

On rappelle que les *Support*$^+$ sont calculés sans l'accès ou le parcours de la base de données.

Les *Support*$^+$ des k-Itemsets sont calculés comme suit :

$$Support^+(\mathrm{Y}) = \min_{|x|=|y|-1} Support \quad (\mathrm{x})$$

- **Etape 2.2 : validation de l'ensemble des k-Itemsets fréquents**

  A la fin de l'étape 2.1, les k-Itemsets candidats sont générés avec des valeurs approximatives de leurs *Support*s alors, un parcours de la base de données est effectué pour le raffinage des résultats et le calcul des *Support*s exacts des k-Itemsets candidats.

L'étape 2 est répétée jusqu'à ce qu'aucun candidat ne puisse être généré.

On présente ici un exemple de déroulement de cet algorithme pour la base de données β et pour un *Support* minimum *Minsup*=2/6 :

Etape 1 : Construction de la structure CountMap

DATABASE β

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

1erParcours

COUNTMAP OPTIMISÉ

A	3	2	3	1	2
B	5	4	0	5	
C	5	1	4		
D	1	0			
E	5				

F_1	*Support*
A	3/6
B	5/6
C	5/6
E	5/6

F_2	*Support*
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

Etape 2 : Extraction des k-Itemsets fréquent

K=3

C_3	*Support*$^+$
ABC	min(2/6,3/6,4/6)=**2/6**
ABE	min(2/6,2/6,5/6)=**2/6**
ACE	min(3/6,2/6,4/6)=**2/6**
BCE	min(4/6,5/6,4/6)=**4/6**

DATABASE
2emeParcourss

F_3	*Support*
ABC	**2/6**
ABE	**2/6**
ACE	**2/6**
BCE	**4/6**

K=4

C_4	*Support*$^+$
ABCE	min(2/6,2/6,2/6,4/6)=**2/6**

DATABASE
3emeParcours

F_4	*Support*
ABCE	**2/6**

En analysant ces résultats, on constate que les valeurs de *Support*s estimées (les *Support*$^+$) sont très proches des *Support*s exacts calculés après la phase de validation. De plus, cette version nécessite trois (03) parcours de la base de données au lieu de quatre pour son homologue Apriori (voire l'exemple section 5.3).

Ces résultats encourageant, nous ont poussé à introduire la deuxième version en retardant la phase de validation jusqu'à la fin de calcul de tous les Itemsets fréquents pour mieux exploiter la notion de *Support*$^+$. Cet algorithme est appelé AprioriCM2.

5.5.3 AprioriCM2 : Version avec validation à posteriori :

Nous avons proposé une autre version de l'algorithme AprioriCM que nous avons appelé AprioriCM à 2 passes (AprioriCM2). Cet algorithme partages la même idée que l'alhotithme AprioriCM mais il utilise une stratégie différente permettant de calculer les Itemsets fréquents en deux parcours de la base de données seulement au lieu de k parcours dans le cas de la version originale d'Apriori et de k-1 dans le cas de AprioriCM-version avec validation par itération (k étant la taille du plus grand Itemset).

```
1.  CM= Gen-CountMap(β,I) ;
2.  L1 =GetFrequentItemsets(CM,1) ;  // calcul des 1-Itemsets fréquents
3.  L2=GetFrequentItemsets(CM,2) ; // calcul des 2-Itemsets fréquents
4.  Pour  (k= 3; Lk-1≠Ø; k++) faire
5.  L+k  = CM-gen(Lk-1);            // la phase de génération-élagage a base de support+
6.  fin ;
7.  Pour  chaque L+k  (k≥3)faire
8.    pour transaction t ∈ β  faire
9.     Ct= sous-ensemble (L+k ,t);    //les candidats contenus dans la transaction t
10.      pour chaque candidates c ∈ Ct faire
11.              c.freq++;
12.     fin;
13.   fin;
14.   Lk={c∈ Ct |c.sup ≥ minsup}
15.  Fin
16.  Résultats= ∪k Lk
```

Fig 1.7 Algorithme AprioriCM2

L'algorithme AprioriCM2 est constitué des étapes suivantes :

- **Etape 1 : Construction de la structure CountMap**

Cette étape est identique à la version précédente de l'algorithme AprioriCM. Un parcours de la base de données est effectué pour construire la structure CountMap. A la fin de ce parcours les Supports des 1-Itemsets et des 2-Itemsets sont calculés et ils sont immédiatement disponibles par un simple accès à CountMap et sans aucun autre parcours de la base de données.

- **Etape 2 : Extraction des k-Itemsets fréquent :**

A cette étape et grâce à la notion de Support$^+$ qu'on a introduit et en utilisant la structure CountMap, on génère itérativement les k-Itemsets candidats. On signale ici que le calcul des k-Itemsets commence à partir de l'itération k=3, les 1-Itemset et les 2-Itemsets sont déjà calculés dans l'étape précédente.

À chaque itération k (k≥3), on construit les k-Itemsets candidats par auto jointure des (k-1)-Itemsets générés dans l'itération précédente. L'ensemble des k-Itemsets candidats est élagué selon la formule (c) (élagage à base de *Support*$^+$) :

Insert into Ck Select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$
From L_{k-1} p, L_{k-1} q
Where p.item$_1$=q.item$_1$, …, p.item$_{k-2}$=q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$
and *Support*$^+$(p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$) ≥ *Minsup*

On rappelle que cela est fait sans l'accès ou le parcours de la base de données. Le calcul de *Support*$^+$ des Itemsets est effectué de façon récursive comme suit :

$$Support^+(\mathrm{Y}) = \begin{cases} \min_{|\mathrm{x}|=|\mathrm{y}|-1} Support \quad (\mathrm{x}), & |\mathrm{Y}| = 3 \\ \min_{|\mathrm{x}|=|\mathrm{y}|-1} Support^+ \ (\mathrm{x}), & |\mathrm{Y}| > 3 \end{cases}$$

Cette étape est répétée jusqu'à ce qu'aucun candidat ne puisse être générée.

- **Etape 3 : raffinage des Itemsets fréquents :**

A ce stade d'algorithme, nous avons calculé les k-Itemset fréquents et comme nous avons signalé auparavant, une phase de validation doit être effectuée pour raffiner cet ensemble d'Itemsets.

Donc, un parcours de la base de données entière est effectué pour recalculer les Supports exacts de ces k-Itemsets. On, signale ici que l'ensemble des 1-Itemsets et 2-Itemsets n'est pas inclu dans ce parcours du fait qu'on a calculé leur Supports exacts à l'étape 1 de l'algorithme via la structure CountMap.

On présente ici un exemple de déroulement de cet algorithme avec les mêmes données de l'exemple précédent. Soit la base de données transactionnelle *D*, on cherche à trouver les Itemsets fréquents pour un *Support* minimum *Minsup*=2/6 :

Etape 1 : Construction de la structure CountMap

DATABASE β

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

1er Parcours

COUNTMAP OPTIMISE

A	3	2	3	1	2
B	5	4	0	5	
C	5	1	4		
D	1	0			
E	5				

F_1	*Support*
A	3/6
B	5/6
C	5/6
E	5/6

F_2	*Support*
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

Etape 2 : Extraction des k-Itemsets fréquent

COUNTMAP OPTIMISÉ

A	3	2	3	1	2
B	5	4	0	5	
C	5	1	4		
D	1	0			
E	5				

C_3	*Support*$^+$
ABC	min(2/6,3/6,4/6)=2/6
ABE	min(2/6,2/6,5/6)=2/6
ACE	min(3/6,2/6,4/6)=2/6
BCE	min(4/6,5/6,4/6)=4/6

C_4	*Support*$^+$
ABCE	min(2/6,2/6,2/6,4/6)=2/6

Etape 3 : raffinage des Itemsets fréquents :

DATABASE β

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

2eme Scan

F_3	*Support*
ABC	2/6
ABE	2/6
ACE	2/6
BCE	4/6

F_4	*Support*
ABCE	2/6

Nous remarquons que les ***Supports***$^+$ estimés sont très proches des Supports exacts trouvés dans la phase de raffinage (Etape3). De plus, cette version nécessite seulement deux accès à la base de données.

5.6 Evaluation de l'algorithme AprioriCM

Nous avons effectué des expérimentations pour évaluer les performances de notre algorithme avec ses deux versions.

Pour cela, nous avons implémenté les deux versions AprioriCM et AprioriCM2 en Java et pour sujet de comparaison, nous avons modifié l'implémentation classique de l'algorithme Apriori en optant à l'utilisation de la structure BitMap pour rapprocher les performances d'Apriori avec les versions AprioriCM et AprioriCM2 qui utilisent une structure de données compact que nous avons introduit (CountMap).

Par conséquent, lors du parcours de la base de données par l'algorithme Apriori, nous utilisons la structure Bitmap qui réside en mémoire central au lieu d'accéder au disque où réside la base de données.

Nous avons utilisé des bases de données expérimentales avec un nombre de transactions de l'ordre de 10^3, 10^5 et 10^6 transactions et un nombre d'items de 8, 12 et 16 items :

DESCRIPTION DES BASE DE DONNEES

Nom de la base de données	#transaction	#items
DBT3333I16	3333	16
DBT100kI16	100000	16
DBTk1000I16	1000000	16

Afin d'étudier les performances de notre algorithme, nous avons effectué un ensemble d'expérimentation en comparant les résultats réalisés par notre algorithme par rapport à l'algorithme Apriori.

Nous avons utilisé un nombre de transactions de l'ordre de 10^3, 10^5 et 10^6 transactions. Nous présentons les résultats obtenus pour les deux versions de notre algorithme :

- **AprioriCM : Version avec validation par itération :**

La figure 1.8 suivante présente les temps d'exécution réalisés par AprioriCM et Apriori pour les bases de données ayant un nombre de transactions de 3333, 100000 et 1000000 :

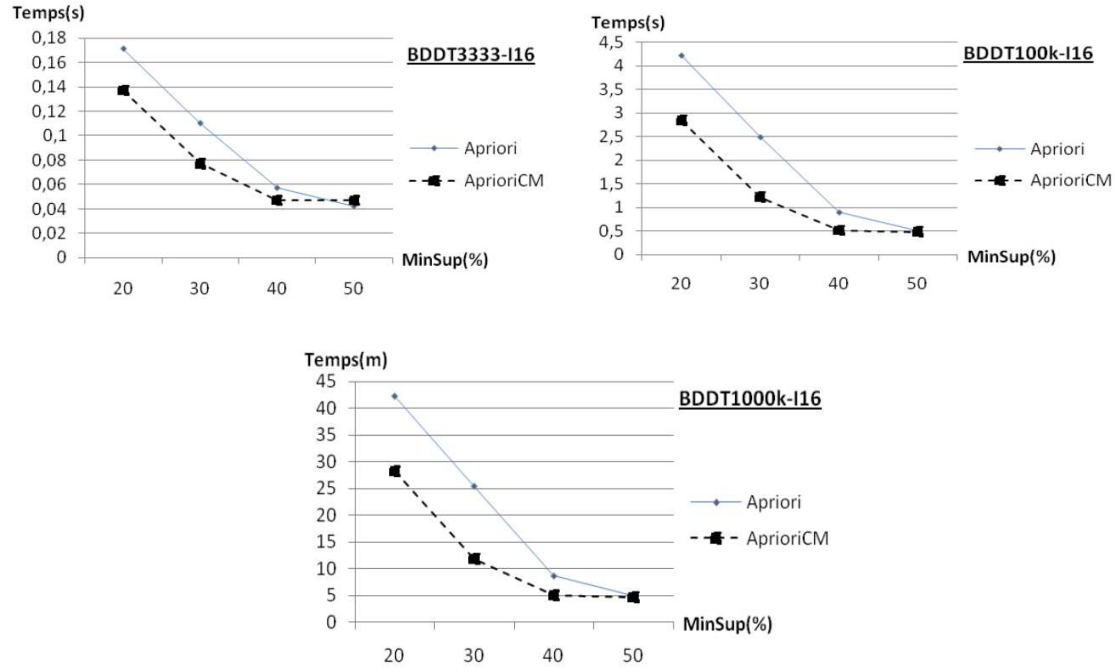


Fig. 1.8 Temps d'exécution : AprioriCM vs Apriori

Les résultats montrent que notre algorithme AprioriCM surpasse l'algorithme Apriori dans les trois bases de données et qu'une meilleure performance est enregistrée par l'algorithme AprioriCM par rapport à Apriori avec l'augmentation de nombre de transactions.

La raison de ces performances est que l'algorithme AprioriCM génère mois d'Itemsets candidats que l'algorithme Apriori. De plus, l'algorithme AprioriCM nécessite k-1 itération au lieu de k itération pour le cas de l'algorithme Apriori.

La figure 1.9 présente le nombre d'Itemsets candidats et le nombre d'Itemsets fréquents générés par les deux algorithmes AprioriCM et Apriori (BDDT1000k-I16).

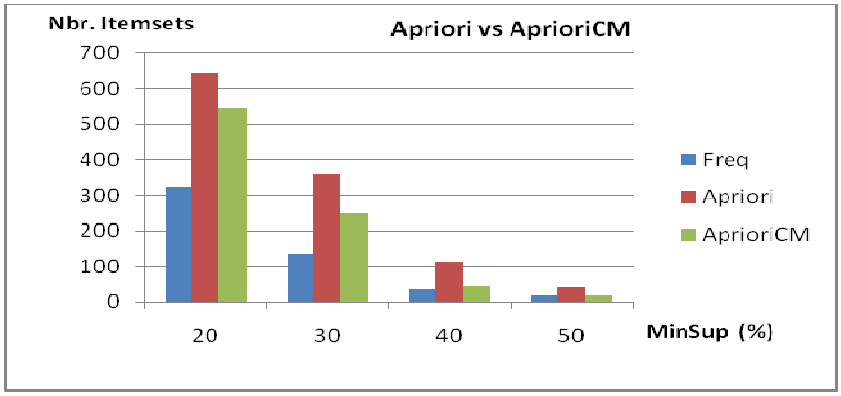


Fig. 1.9 Nombre d'Itemsets : AprioriCM vs Apriori

Ces résultats montrent que l'algorithme AprioriCM généré moins d'Itemsets candidats (colonnes vertes) que l'algorithme Apriori (colonnes rouges).

Cela est à cause de la technique d'élagage à base de Support^{+} qui permet d'éliminer un nombre plus grand d'Itemsets candidats que l'algorithme Apriori.

- **AprioriCM2 : Version avec validation à postériori :**

Nous présentons ici les performances réalisées par l'algorithme AprioriCM2 (Version avec validation à postériori) par rapport à l'algorithme Apriori et cela en variant le nombre de transactions de la base de données.

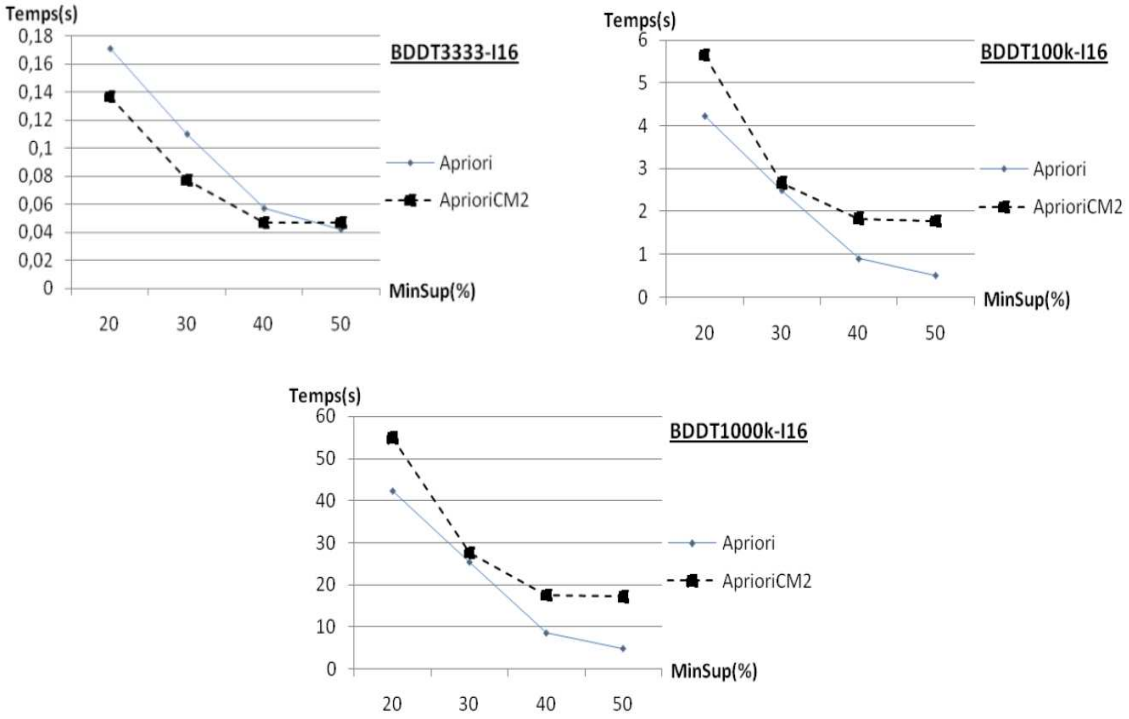


Fig. 1.10 Temps d'exécution : AprioriCM2 vs Apriori

Deux cas de figures se présentent, pour une base de données avec un nombre de transaction relativement réduit (BDDT3333I16), nous constatons une meilleure performance de l'algorithme AprioriCM2 par rapport à l'algorithme Apriori vu que l'algorithme AprioriCM2 nécessite que 2 parcours de la base de données et que cette dernière contient un nombre de transaction relativement réduit et donc une phase de raffinage mois coûteuse.

Pour les bases de données avec un grand nombre de transactions (BDDT100kI16 et BDDT1000kI16), l'algorithme AprioriCM2 est mois performante fasse à l'algorithme Apriori et cela à cause du retardement de la phase de raffinage des Itemsets candidats. Cela dit un nombre de candidats supplémentaire est généré par l'algorithme AprioriCM2 et par conséquent une phase de raffinage plus coûteuse.

La figure 1.11 présente le nombre d'Itemsets candidats et le nombre d'Itemsets fréquents générés par les deux algorithmes AprioriCM2 et Apriori.

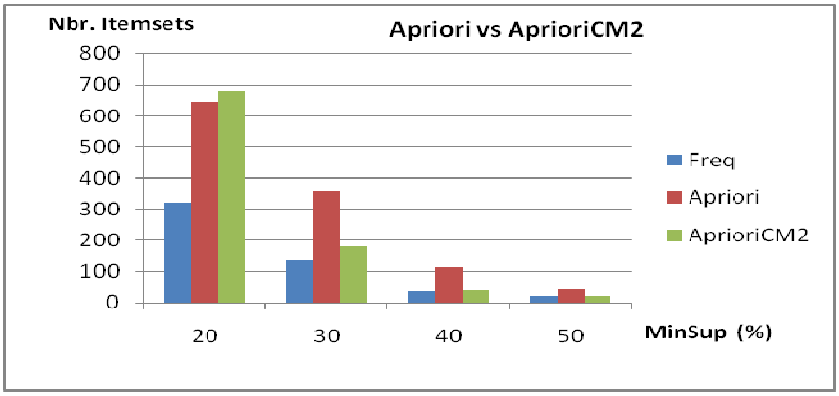


Fig. 1.11 Nombre d'Itemsets : AprioriCM2 vs Apriori

Ces résultats montrent que l'algorithme AprioriCM2 généré plus d'Itemsets candidats (colonnes vertes) que l'algorithme Apriori (colonnes rouges) est cela à cause du retardement de la phase de raffinage. L'algorithme Apriori exécute la phase de raffinage (calcul des Supports exactes) à chaque itération contrairement à l'algorithme AprioriCM2 qui retarde cette phase jusqu'à la fin de l'algorithme.

- **Synthèse des résultats**

Les expérimentations menées dans un contexte séquentiel pour l'évaluation de la technique d'élagage à base de Support$^+$ montrent l'intérêt de cette technique dans la réduction du nombre d'Itemsets candidats générés et donc le coût de calcul des Itemsets fréquents d'un coté et la réduction du nombre d'accès à la base de données pour le calcul des Supports exacts des Itemsets candidats d'un autre coté.

Deux stratégies d'exploitation de la technique d'élagage à base de Support$^+$ sont implémentées et évaluées à savoir l'algorithme AprioriCM (validation par itération) et AprioriCM2 (validation à postériori).

Les expérimentations ont montré l'efficacité et la supériorité de la version AprioriCM par rapport à Apriori et la faiblesse de la version AprioriCM2 à cause du retardement de la phase de raffinage qui devient son inconvénient.

6 Conclusion :

Plusieurs algorithmes séquentiels efficaces d'extraction de règles d'associations ont été proposés dans les dernières années avec une variété de détails d'implémentation. Ils se distinguent par leurs stratégies d'énumération et d'élagage de l'espace de recherche des Itemsets ainsi que par la méthode de calcul du Support. De plus, quelques algorithmes utilisent des structures de données sophistiquées comme la table de hachage pour plus de performance.

Nous avons introduis une nouvelle stratégie pour le calcul des Itemsets fréquents. Les performances réalisées par notre algorithme AprioriCM par rapport à la version classique d'Apriori démontre l'utilité de notre approche.

La version ApriroiCM2 étant moins performante que Apriori et AprioriCM, mais nécessite seulement deux accès à la base de données et donc une réflexion est fait pour l'exploitation de cette version dans un contexte distribuées ce qui implique moins de communication/synchronisation dans le cas où la base de données est distribuée.

1 Introduction

En raison de l'organisation logistique des entités qui collectent les données - soit des sociétés privées ou des établissements publics - les données sont souvent distribuées à l'origine. Les succursales (branches) d'une banque ou les bureaux de recensement dans un pays sont tous des exemples de données collectées d'une façon distribuée. Les données sont typiquement trop grandes pour être réunies dans un seul site ou bien pour des questions de confidentialité, elles peuvent seulement être déplacées à travers un ensemble limité de sites alternatifs. Dans d'autres cas, les données sont produites localement, mais en raison de leur volume énorme ne peuvent pas être stockées dans un seul site donc elles sont déplacées immédiatement après leur production à d'autres emplacements de stockage, distribuée typiquement à l'échelle géographique. Ainsi, le DataMining Distribué a attiré beaucoup d'attention dans la communauté de DataMining.

Le DataMining distribué se réfère à l'extraction des connaissances à travers des sources de données distribuées. Les algorithmes de DataMining doivent se confronter aux possibilités limitées de déplacement/réplication de ces données en raison de politiques spécifiques ou des raisons techniques. Dans le DataMining centralisé, le principal souci pour l'efficacité d'un algorithme de DataMining est son temps d'entrée-sortie et/ou son temps CPU. Dans un environnement distribué, on doit considérer aussi le coût de communication. Le coût de communication est déterminé par la largeur de bande de réseau et le nombre de messages qui sont envoyés à travers le réseau.

De plus, l'intégration des résultats n'est pas simplement la réunion des résultats de tous les sites. Un modèle (motif) intéressant dans une base de données locale peut ne pas être intéressant globalement. Par exemple, un Itemset fréquent à un site local peut être non fréquent globalement. Puisque le but de DataMining distribué est la recherche des modèles (motifs) globaux intéressants. Les modèles et leurs propriétés doivent être rassemblés de tous les sites et vérifiés globalement pour être intéressants.

Il y a beaucoup d'autres facteurs qui doivent être considérés dans le DataMining distribué, comme la sécurité, la confidentialité, l'autonomie de base de données locale, la topologie de réseau et le schéma de communication et la charge de travail de chaque site.

Nous nous intéressons dans ce chapitre au problème des règles d'associations distribuées. Comme nous avons signalé dans le chapitre précédent, l'accent est mis sur l'étape d'extraction des Itemsets fréquents. Nous introduisons la formulation du problème d'extraction distribuée d'Itemsets fréquents. Nous présentons un état d'art détaillé des algorithmes distribués d'extractions des Itemsets fréquents. Par la suite, nous présentons notre approche pour l'extraction des Itemsets distribués.

2 Le problème d'extraction des Itemsets fréquents distribués

L'extraction des règles d'association se fait en général en deux étapes. La première consiste à extraire l'ensemble des Itemsets fréquents ou motifs fréquents ; les règles d'association sont générées, dans la deuxième étape à partir de ces motifs. L'extraction des Itemsets fréquents dans les bases de données est l'étape primordiale et la plus coûteuse dans l'extraction des règles d'association, car le nombre d'Itemsets fréquents est exponentiel par rapport au nombre d'items dans la base de données.

- **Formulation du problème :**

Soit *I* l'ensemble d'items et *DB* la base de données des transactions, où chaque transaction a un identificateur unique (tid) et contient un ensemble d'items appelés Itemset. Le Support d'un Itemset Y, noté Supp(Y), est le nombre de transactions dans lesquelles cet Itemset apparaît comme un sous-ensemble.

Dans un contexte distribué, la base de données *DB* est partitionnée en $\{DB_1, DB_2, \ldots, DB_p\}$ et distribuée à travers les *p* sites $S_1, S_2, \ldots, S_p$.

Soit *D* la taille de la base de données *DB* et d_i la taille de la partition DB_i. Pour un Itemset Y donné, soit Supp(Y) et $Supp_i(Y)$ le Support de Y dans *DB* et DB_i respectivement. Dans ce cas, Supp(Y) est appelé le Support "global" et $Supp_i(Y)$ le Support "local" de Y dans le site S_i.

Pour un Support minimal donné *Minsup*, l'Itemset Y est globalement fréquent si $Supp(Y) \times D \geq Minsup \times D$. De même, l'Itemset Y est localement fréquent dans le site S_i Si $Supp_i(Y) \times d_i \geq Minsup \times d_i$.

Ainsi, le problème d'extraction des règles d'association dans la base de données distribuée se réduit à trouver tous les Itemsets globalement fréquents et de générer ensuite les règles d'association correspondantes.

3 Les Algorithmes d'extraction des Itemsets fréquents distribués :

Les algorithmes parallèles et distribués actuels sont basés sur l'algorithme séquentiel Apriori. Deux paradigmes de parallélisme sont employés : le parallélisme de données et le parallélisme de tâches [CHE 96a]. Les deux paradigmes diffèrent selon la distribution de l'ensemble de candidats à travers les sites ou non. Dans le parallélisme de données, chaque site dénombre le même ensemble de candidats tandis que dans le parallélisme de tâches, l'ensemble de candidats est divisé et distribué à travers les sites et chaque site dénombre un ensemble différent de candidats.

3.1 Algorithmes de parallélisme de données :

Dans ce type de paradigme, chaque site dénombre le même ensemble de candidats. Les candidats sont dupliqués sur tous les sites et la base de données est distribuée à travers ces sites. Chaque site est responsable de calculer les Supports locaux de tous les candidats dans sa propre partition de base de données. Ensuite, les sites calculent les Supports globaux (les Supports totaux des candidats dans la base de données entière) par l'échange des Supports locaux (la Réduction Globale). Ensuite, les Itemsets fréquents sont calculés indépendamment par chaque site. Ce paradigme est illustré dans la Figure 2.1 :

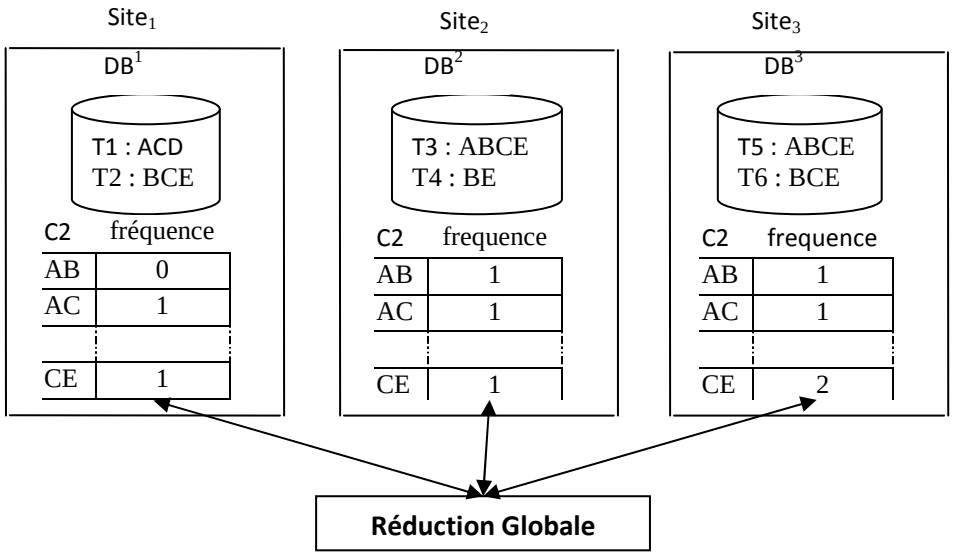


Fig. 2.1 Paradigme parallélisme de données

Les transactions sont partitionnées à travers les trois sites. Par exemple le site 1 a les deux transactions (T1, T2), le site 2 a les transactions (T3,T4) et le site 3 a les transactions (T5,T6). La figure 2.1 montre les Supports locaux après le balayage des bases de données locales. Les Itemsets candidats dans le deuxième balayage sont dupliqués sur chaque site.

Plusieurs algorithmes existent pour ce paradigme. Ces algorithmes sont : CD [AGR 96], PDM [PAR 95], DMA [CHE 96b], CCPD [ZAK 96], NPA [SHI 96], FDM [CHE 96a], FPM [CHE 98]. Ces algorithmes parallèles diffèrent par les techniques d'élagage des nouveaux candidats employés ou par les stratégies de calcul des Supports des candidats employés.

- **CD (Count Distribution)**

L'algorithme CD [AGR 96] est une simple parallélisation d'Apriori. Cet algorithme se base sur une répartition "aléatoire" des données en effectuant un découpage horizontal des données. La base de données *DB* est divisée en {DB_1, DB_2 …, DBp} et elle est distribuée à travers les sites. L'algorithme est constitué essentiellement de trois étapes. Dans l'étape 1, les Supports locaux des candidats C_k dans la partition locale DB_i de chaque site i sont calculés. Dans l'étape 2, les sites se synchronisent et chaque site échange les Supports locaux de tous les candidats avec tous les sites pour calculer les Supports globaux des candidats C_k. Dans l'étape 3, les Itemsets globalement fréquents L_k sont identifiés et les candidats de taille k+1

sont générés indépendamment sur chaque site à partir de l'ensemble L_k. L'algorithme CD répète les étapes 1 - 3 jusqu'à ce qu'aucun autre candidat ne peut être généré.

- **PDM (Parallel Data Mining)**

L'algorithme PDM [PAR 95] est une modification de l'algorithme CD par l'utilisation d'une technique de hachage directe pour l'élagage de quelques candidats dans le passage suivant. Dans le premier passage, en plus du dénombrement de tous les 1-Itemsets, l'algorithme PDM maintient une table de hachage pour stocker les Supports de 2-Itemsets. Entre chaque itération, des communications similaires à celles de l'algorithme CD (une émission "tous à tous" des Supports locaux) sont utilisées pour le calcul des Supports globaux. Cependant, cet échange direct des Supports peut être coûteux, les auteurs emploient une méthode optimisée qui permet l'échange seulement des cellules qui sont sûr d'être fréquentes. Cette méthode exige deux tournées de communication. L'algorithme PDM a plusieurs limitations. L'algorithme parallélise la génération de candidat aux dépens d'une émission "tous à tous" pour construire l'ensemble entier des candidats (Les coûts de communication peuvent rendre cette parallélisation inefficace). De plus, les auteurs ont présenté seulement les résultats de simulation sur une machine IBM-SP2 de type mémoire distribuée, et donc l'évaluation pratique de l'impact de leurs optimisations est difficile.

- **DMA (Distributed Mining Algorithm)**

L'algorithme DMA [CHE 96b] est une parallélisation de l'algorithme Apriori (plus précisément de son amélioration DHP). L'algorithme introduit en plus des techniques d'élagage de candidats et de réduction de message de communication. L'algorithme utilise les Itemsets lourds (un Itemset localement fréquent dans une partition et globalement fréquent dans la base de données entière) sur chaque site pour générer les candidats à partir de ces Itemsets fréquents lourds. Cette méthode permet de réduire le nombre de candidats générés. Soit par exemple les items lourds A et B respectivement sur le site 1 et 2, c'est-à-dire A est globalement fréquent et localement fréquent seulement sur le site 1, B est globalement fréquent et localement fréquent seulement sur le site 2. L'algorithme DMA ne génère pas l'Itemset AB comme un 2-Itemset candidat contrairement à l'algorithme CD. Les Supports sont échangés lors des communications/synchronisées entre chaque itération et au lieu de la diffusion des Supports locaux de tous les candidats comme dans le cas de CD, DMA envoie seulement les Supports locaux à un site élu en réduisant ainsi la taille des messages de $O(p^2)$ à $O(p)$ (p est le nombre de sites).

- **NPA (Non Partitioned Apriori)**

L'algorithme NPA est proposé par [SHI 96] à base d'Apriori. NPA est essentiellement le même que l'algorithme CD (Count Distribution) avec la seule différence que la réduction globale a lieu sur un site maître.

- **FDM (Fast Distributed Mining)**

L'algorithme FDM [CHE 96a] est basé sur l'algorithme CD et propose de nouvelles techniques pour réduire le nombre de candidats et le coût de communication de CD. L'algorithme FDM suppose que chacun des Itemsets globalement fréquents doit être localement fréquent sur un site. Ainsi un site i génère les Itemsets candidats (noté GL*i)* seulement à partir des Itemsets globalement fréquents et localement fréquents dans le site i. Par exemple, soit F_1={A, B, C, D, E} l'ensemble de tous les 1-Itemsets fréquents et GL_1 = {A, B, C} et GL_2 = {C, D, E} les ensembles d'Itemsets globalement et localement fréquents respectivement dans le site 1 et le site 2. Le site 1 considère donc seulement les candidats CG_1 = {AB, AC, BC} et le site 2 les candidats CG_2 = {CD, CE, DE}. Ainsi FDM génère seulement six Itemsets candidats (CG_1 et CG_2) au lieu de 28 candidats dans le cas de l'algorithme CD.

L'algorithme FDM suggère aussi trois optimisations : élagage local, élagage global et polling de Supports. L'algorithme FDM-LP emploie l'élagage local et le polling de Support, chaque site i génère les candidats CGi et assigne un site élu pour chaque candidat. Ensuite, chaque site i calcule le Support local de tous les candidats assignés à ce site et applique la technique d'élagage local : enlever tout Itemset Y qui n'est pas localement fréquent sur le site i (si Y est globalement fréquent, alors il doit être localement fréquent sur d'autres sites).

Une autre optimisation consiste à l'utilisation d'un site élu pour le calcul de Support. Chaque site élu calcul les Supports globaux de tous les candidats qui sont assignés à lui en récupérant les Supports locaux de tous les autres sites. Le site élu diffuse ensuite les Supports globaux à tous les autres sites. À la fin, chaque site a l'ensemble d'Itemsets globalement fréquents et donc une nouvelle itération peut commencer. On rappelle que CD diffuse les Supports locaux de tous les candidats à chacun des sites, tandis que FDM les envoient seulement à un site élu par candidat. Ainsi, FDM exige encore moins de communication et l'élagage local le diminue encore plus. Une autre optimisation que les auteurs suggèrent est l'élagage global. Au lieu d'envoyer seulement les Supports globaux des fréquents à la fin de l'itération k-1, ils envoient aussi leurs Supports locaux de chaque partition. Pour l'itération suivante k, si Y est un candidat, les Supports locaux de toutes ses (k-1)-sous-ensembles sont disponibles. Les auteurs ont évalué FDM sur un groupe de six postes de travail connectés avec un réseau local Ethernet 10Mo. Les expériences effectuées évaluent seulement l'élagage local avec l'élection de site de calcul de Support et les résultats ont montré une réduction de 75% à 90% de la taille de l'ensemble de candidats sur chaque site et une réduction de 85% à 90% dans la taille des messages par rapport à CD.

- **FPM (Fast Parralle Mining)**

[CHE 98] a récemment proposé une version parallèle de FDM, appelé FPM (Fast Parallel Mining). L'algorithme FPM génère moins de candidats et conserve les étapes d'élagage local et global de FDM. Le problème avec le mécanisme d'élection de FDM est qu'il exige deux tournées de messages dans chaque itération : une tournée pour calculer les Supports globaux et une pour diffuser les fréquents. Ce schéma à deux tournées peut dégrader la performance de

l'algorithme et donc l'algorithme FPM émet simplement les Supports locaux à tous les sites. L'aspect le plus intéressant de ce travail est la métrique définie pour l'asymétrie de données (la distribution d'Itemsets entre les diverses partitions). Les expériences sur un IBM SP2 à 32-nœuds indiquent que FPM peut surpasser CD par un facteur de 3 pour des ensembles de données avec une très haute asymétrie et par un facteur de 1.5 pour des données de basse asymétrie.

- **ODAM (Optimized Distributed Association Mining)**

ODAM [ASH 04] est un algorithme distribué pour les bases de données géographiquement distribuées. L'algorithme réduit le coût des communications en réduisant au minimum le coût de génération des candidats. L'algorithme ODAM calcule d'abord les Supports des 1-Itemsets sur chaque site de la même manière que fait l'algorithme séquentiel Apriori. L'algorithme diffuse ensuite cet ensemble d'Itemsets aux autres sites et les 1-Itemsets globaux fréquents sont identifiés.

Par la suite, l'algorithme ODAM élimine tous les 1-Itemsets globaux non fréquents de l'ensemble des transactions de la base de données (les données initialement contiennent les items fréquents et non fréquents). En insérant la nouvelle transaction (transactions sans les 1-Itemsets non fréquents), l'algorithme vérifie si cette transaction existe déjà dans la mémoire et donc nous incrémentons le compteur associé à cette transaction par un, autrement nous l'insérons comme une nouvelle transaction avec un compteur égal à un. Ceci réduit les tailles des transactions (le nombre d'items des transactions) et le nombre de transaction de la base de données (les transactions identiques sont réduites à une seule transaction à laquelle est associée un compteur).

En même temps, chaque site génère les 2-Itemsets candidats à partir d'ensemble des 1-Itemsets globaux fréquents et calcule leurs Supports dans chaque site et les envoient à un site, appelé le site récepteur pour le calcul des 2-Itemsets fréquents globaux. L'algorithme réitère sur les transactions en mémoire principale en générant les candidats du niveau suivant et en diffusant les Itemsets fréquents globaux calculés après chaque itération.

- **DDM (Distributed Decision Minner)**

L'algorithme DDM [SCH 01] est un algorithme pour l'extraction des Règles d'Association distribués basées sur Apriori. Les candidats dans DDM sont générés comme dans FDM (élagage local) par chaque site. Ensuite, les sites exécutent un Protocole de Décision Distribué pour identifier lesquels des candidats sont fréquents et lesquels ne le sont pas. L'algorithme DDM diffère de FDM dans le fait que le protocole de DDM permet à certains sites de décider de publier les fréquences locales d'un candidat et non d'un autre. Le protocole est dirigé par deux hypothèses (publique est privé) qui sont maintenues sur chaque candidat : La première, appelé l'hypothèse publique, est que chaque site assume que la fréquence globale d'un Itemset est égale à la moyenne des fréquences locales publié pour cet Itemset jusqu'ici (ou nulle si rien n'a été publié) ; pour la deuxième, appelé l'hypothèse privée, chaque site assume que ces fréquences locales sont partagées par tous ceux qui n'ont pas publié leur fréquence

locale pour cet Itemset candidat. Si un site trouve que l'hypothèse publique et privée sur un Itemset ne concorde pas (l'un prévoit que l'Itemset est fréquent tandis que l'autre prévoit qu'il n'est pas fréquent), le site publie alors la fréquence locale de cet Itemset. L'algorithme DDM est jusqu'ici plus efficace en communication, en scalabilité, et en résistance à l'asymétrie des données.

- **Discussion :**

Le parallélisme de données offre une communication plus simple et moins surchargée : seuls les Supports locaux des Itemsets candidats sont échangés à chaque itération. Dans ce type d'algorithmes, on doit échanger les informations calculées sur les différents sites, d'où beaucoup de communications que l'on peut considérer assez courtes ; dans le meilleur des cas, on ne communique que des entiers (les Supports).

L'algorithme CD étant l'algorithme de base. C'est une simple parallélisation d'Apriori. Cet algorithme minimise les communications, parce que seuls les Supports sont échangés entre les sites. Les algorithmes PEAR, PDM, NPA, FDM, FPM et CCPD sont tous similaires et proposent des améliorations à CD en employant soit les techniques de hachages, les techniques d'élagage des candidats. L'algorithme NPA est essentiellement le même que CD, sauf que la réduction globale a lieu sur un site maître. Les algorithmes FDM et DMA se résultent dans le petit nombre de candidats et les petites tailles des messages par rapport à CD. FDM se base sur CD et propose de nouvelles techniques pour réduire le nombre de candidats considérés pour le Support par l'élagage local, l'élagage global et l'élection de site de calcul de Support. De plus, CD diffuse les Supports locaux de tous les candidats à chacun des sites, tandis que FDM les envoie seulement à un site élu par candidats indépendants du site ou se trouve ce candidat. Ainsi, FDM exige encore moins de communication et l'élagage local les diminue encore plus. L'algorithme FPM, la version parallèle de FDM génère moins de candidats et conserve les étapes d'élagage local et global et au lieu du mécanisme d'élection de FDM, FPM émet simplement les Supports locaux à tous les sites car le mécanisme d'élection exige deux tournées de messages dans chaque itération (une pour calculer les Supports globaux et une pour diffuser les fréquents) ce qui peut dégrader la performance de l'algorithme. L'algorithme ODAM proposé à base de CD permet de réduire les tailles des transactions en éliminant les items non fréquents dans les transactions et en fusionnant plusieurs transactions identiques dans une seule transaction. Quant à l'échange de message, au lieu d'utilisation du broadcaste comme le cas de CD ou le site élu dans le cas de FDM, l'algorithme ODAM juste envoie toute l'information locale à un site, appelé le récepteur. Le récepteur calcule alors les fréquents globaux et les renvoie en retour. L'algorithme ODAM surpasse FDM et CD, et il se montre plus scalable car il génère moins de messages. L'algorithme DDM se montre plus scalable que CD et FDM respectivement au nombre de sites participants et au Support minimal. De plus, DDM est plus efficace dans la présence de l'asymétrie et les données.

3.2 Algorithmes de parallélisme de tâches:

L'ensemble de candidats est partitionné et distribué à travers les sites ainsi que la base de données. Chaque site est responsable de maintenir seulement les Supports globaux d'un sous-ensemble de l'ensemble des candidats. Cette approche exige à chaque itération deux sous-phases de communication. Dans la première sous-phase, chaque site envoie sa partition de la base de données à tous les autres sites. Dans la seconde, chaque site diffuse les Itemsets fréquents qu'il a trouvé à tous les autres sites pour calculer les candidats de l'itération suivante. Ce paradigme est illustré par la Figure 2.2 :

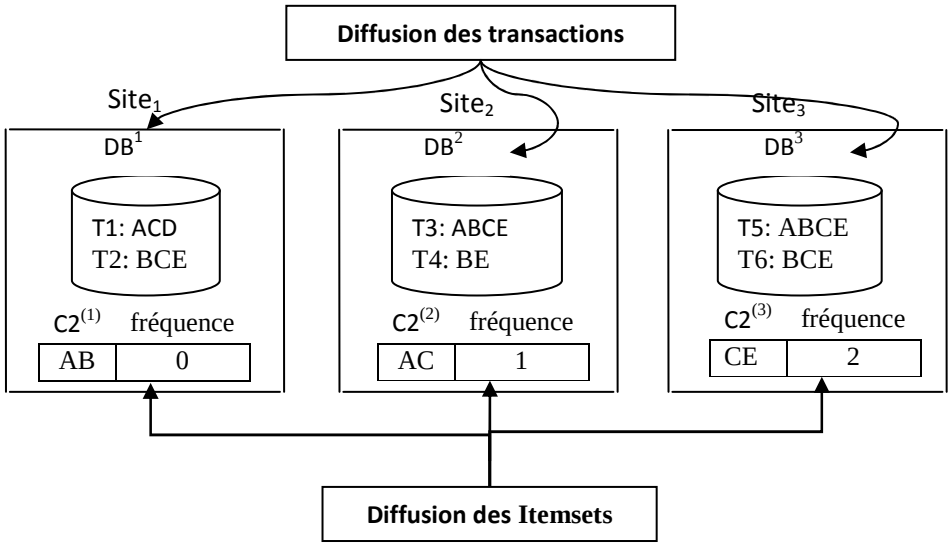


Fig. 2.2 Paradigme parallélisme de tâches

Les transactions sont partitionnées comme dans le cas du parallélisme de données. Les candidats sont partitionnés à travers les 3 sites. Après le balayage de la base de données locale et l'émission des partitions des bases de données des autres sites, on obtient le Support global de chaque candidat.

- **DD (Data Distribution) :**

L'algorithme DD [AGR 96] se base sur une répartition "aléatoire" des données : les candidats sont partitionnés en des ensembles disjoints et ils sont distribués sur tous les sites dans une manière round robin. L'algorithme est composé de trois étapes. Dans l'étape 1, chaque site parcourt sa partition locale pour y obtenir les Supports locaux des candidats qu'ils lui ont été distribués (assignées). Dans l'étape 2, chaque site diffuse sa partition de base de données aux autres sites et reçoit les autres partitions des autres sites (communication tous à tous), chaque site parcourt ensuite les partitions reçues pour obtenir les Supports globaux de la base de données entière. Dans l'étape 3, chaque site calcule les fréquents de son ensemble de candidats, il les échange avec tous les autres sites pour obtenir tous les fréquents et il génère ensuite les candidats, les partitionne et les distribue sur tous les sites. Ces étapes continuent jusqu'à ce qu'il n'y a plus de candidats à générer. L'algorithme SPA proposé par [SHI 96] est similaire à DD sauf que SPA effectue la réduction globale sur un site maître.

- **IDD (Intelligente Data Distribution) :**

L'algorithme IDD [HAN 97] est une amélioration de l'algorithme DD. Premièrement, au lieu d'un partitionnement des Itemsets candidats selon le principe round robin, l'algorithme IDD partitionne les Itemsets candidats à travers les sites en se basant sur le premier item des Itemset candidats, c'est-à-dire les Itemsets candidats avec le même premier item seront mis dans la même partition. Donc, avant le traitement d'une transaction, chaque site doit s'assurer qu'elle contient les préfixes(les premiers items) appropriés qui sont assignés à ce site. Sinon, la transaction peut être écartée. La base de données entière est toujours communiquée, mais une transaction ne pourrait pas être traitée si elle ne contient pas les items appropriés. Ce qui réduit le calcul redondant dans l'algorithme DD (dans l'algorithme DD, chaque site doit vérifier tous les sous-ensembles de chaque transaction).

De plus, l'algorithme IDD assure un équilibre de charge dans la distribution des Itemsets candidats en calculant d'abord pour chaque item le nombre d'Itemsets candidats commençant par l'item en question et assignant en suite les items aux partitions des Itemsets candidats de telle sorte que le nombre d'Itemsets candidats dans chaque partition soit égal.

En outre, l'algorithme IDD commute pour l'algorithme CD (Count Distribution) si les Itemsets candidats peuvent être tenus en mémoire.

En fin, l'algorithme emploie une communication asynchrone de point à point entre les voisins dans une architecture en anneau au lieu du broadcast pour réduire les coûts de la communication et améliorer les performances.

- **HPA (Hash-based Parallel mining of Association rules):**

HPA [SHI 96] emploie une technique de hachage pour distribuer les candidats à travers les différents sites. Chaque site génère l'ensemble de candidats fréquents et applique une fonction de hachage pour déterminer le site (hébergeur) pour chaque candidat. Si un site est bien le site hébergeur de ce candidat, alors le candidat est inséré dans l'arbre de hachage local; sinon sera écarté. Pour le calcul de Support et au lieu de déplacer les partitions de la base de données entre les sites, l'algorithme HPA déplace les sous-ensembles d'items des transactions à leurs sites de destination par la même technique de hachage. Donc seul un sous-ensemble d'items d'une transaction est déplacé à un site au lieu de n sous ensembles.

L'algorithme HPA a été amélioré par une version appelée HPA-ELD [SHI 96] (HPA-Extremely Large Duplication) pour traiter le problème de l'asymétrie dans les données. Bien que les Itemsets candidats puissent être distribuées équitablement entre les sites, certains Itemsets candidats sont plus fréquents que d'autres et par conséquent certains sites seront plus surchargés que d'autre. Donc l'algorithme HPA-ELD propose de dupliquer certains Itemsets candidats dans chaque site.

- **Discussions :**

Le paradigme de parallélisme de tâche a été initialement proposé pour utiliser efficacement la mémoire globale principale d'un ordinateur parallèle. Les algorithmes partitionnent et distribuent les ensembles de candidats disjoints entre les sites à chaque itération, donc ils utilisent la mémoire globale principale de tous les sites. Cependant, pour calculer le Support, chaque site doit parcourir la base de données entière (sa partition locale et toutes les partitions distantes) dans toutes les itérations. Ce type d'algorithmes fait "tourner" les transactions sur chaque site. Dans l'algorithme de base DD (Data Distribution), La partition de la base de données sur chaque site est émise à tous les autres sites (émission "tous à tous"). Le coût total pour le mouvement de la base de données est O(p2) où p est le nombre de sites impliqués. Les algorithmes SPA, IDD sont similaires à DD. L'algorithme IDD suppose une architecture en anneau et la communication est faite simultanément entre les voisins, donc le coût total est d'ordre de O(p). De plus, IDD effectue le partitionnement des candidats à base de préfixe pour assurer que les transactions contiennent les préfixes appropriés avant leur traitement sinon, la transaction peut être écartée.

3.3 Autres parallélismes :

Il y a quelques autres algorithmes qui ne peuvent pas être classifiés dans les deux paradigmes à proprement parler. Bien qu'ils partagent des idées semblables avec les deux paradigmes, ils ont des particularités distinctes. Ces algorithmes incluent Candidate Distribution [AGR 96], SH [HAR 98] et HD (Hybrid Distribution) [HAN 97].

- **CaD (Candidate Distribution)**

L'algorithme CaD [AGR 96] est une sorte d'hybridation des deux paradigmes CD (Count Distribution) et DD (Data Distribution) qui essaye de réduire les coûts de la synchronisation/communication. L'algorithme se base sur un découpage horizontal des données, mais également sur une distribution des candidats de façon à assurer une cohérence entre les Itemsets qu'on calcule et les transactions qu'on utilise pour effectuer ces calculs. Ainsi, il faut effectuer une redistribution des données entre chaque itération. Dans l'itération k, l'algorithme divise les Itemsets fréquents L_{k-1} entre les sites de façon que chaque site puisse générer un ensemble unique disjoint de candidats indépendants des autres sites. En même temps, la base de données est distribuée pour qu'un site puisse calculer les Supports des candidats qu'il a générés indépendamment des autres. Notez que selon la qualité de la division des candidats, des parties de la base de données peuvent être répliqués sur plusieurs sites. La distribution des Itemsets est faite par regroupement basé sur leurs préfixes communs. Après cette distribution des candidats, chaque site continue indépendamment de calculer les Supports de candidats qu'on a assignés à ce site en employant seulement la partition de base de données locale.

- **SH (Skew Handling) :**

Dans SH [HAR 98], les Itemsets candidats ne sont pas générés à priori à partir des Itemsets fréquents précédents mais ils sont générés indépendamment par chaque site instantanément en parcourant sa partition local de la base de données. Dans l'itération k, chaque site génère et calcule les k-Itemsets dont tous les (k-1)-sous-ensembles sont globalement fréquents en vérifiant un bitmap de tous les (k-1)-Itemsets globalement fréquents. À la fin de chaque itération, tous les sites échangent les k-Itemsets entre eux pour obtenir les k-Itemsets fréquents globaux. Les k-Itemsets fréquents sont alors identifiés sur chaque site et le bitmap des Itemsets fréquents est mis à jour.

En cas du déséquilibre de charge, les transactions sont migrées vers les sites non surchargés et en cas où la mémoire principale insuffisante, les k-Itemsets courants sont triés et embobinés au disque et de nouveaux k-Itemsets sont générés et calculés pour le reste de la partition de la base de données.

SH semble être basé sur un algorithme différent d'Apriori, mais c'est très proche d'Apriori. D'abord, il est itératif comme Apriori, c'est-à-dire, seulement à la fin d'une itération où les nouveaux candidats de taille supérieur sont générés. La différence avec Apriori réside dans le moment où les candidats sont générés : SH génère les candidats instantanément à partir des transactions tandis qu'Apriori les génèrent a priori à la fin de chaque itération. Deuxièmement, les candidats générés par SH sont exactement les mêmes que ceux généré par Apriori dans le cas où la base de données est équitablement distribuée. Cependant, les candidats seront différents si la base de données est extrêmement asymétrie. Par exemple, si A et B n'apparaît jamais ensemble dans la partition i, alors SH ne génère pas AB comme un candidat dans la deuxième passe sur le site i. Mais si A et B apparaît ensemble une fois, alors AB sera généré comme candidat.

Donc, nous pouvons classifier SH dans le paradigme de parallélisme de données avec le traitement de l'asymétrie et le traitement de l'insuffisance de la mémoire principale.

- **HD (Hybrid Distribution)**

L'algorithme HD [HAN 97] est une solution hybride combinant IDD et CD. Les *p* sites sont divisés en *G* groupes de tailles égaux, où chaque groupe est considéré comme un super-site. L'algorithme IDD est employé au sein des groupes et l'algorithme CD entre les groupes. L'algorithme HD suppose que les *p* sites sont arrangés dans une grille à deux dimensionnes de *G* lignes et de *p/G* colonnes. La base de données est divisée horizontalement entre les *G* groupes et les candidats C_k sont divisés entre les *P/G* sites dans un groupe. On peut exécuter n'importe quel algorithme de distribution de données indépendamment le long de chaque colonne de la grille et les Supports globaux de chaque sous-ensemble de C_k sont obtenus par la fusion des Supports le long de chaque ligne de la grille comme dans le paradigme de parallélisme de données.

L'architecture supposée en grille peut être vue comme une généralisation des deux paradigmes : Si le nombre de colonnes dans la grille est égal à 1, l'algorithme se ressemble

aux algorithmes du paradigme de parallélisme de tâche et si le nombre de lignes est égal à 1, l'algorithme se ressemble aux algorithmes du paradigme de parallélisme de données.

Dans HD, le coût de la communication de la base de données est réduit du fait que les partitions sont déplacées seulement le long des colonnes de grille au lieu de la grille entière (tous les sites). HD peut aussi commuter automatiquement à l'algorithme CD dans les itérations ultérieures pour réduire davantage le coût de la communication. En plus, HD ajuste les groupes dynamiquement pour chaque passage.

- **PAR-Eclat, PAR-MaxEclat**

Un ensemble d'algorithmes (PAR-Eclat, PAR-MaxEclat, PAR-Clique et PAR-MaxClique) qui emploient un partitionnement de la base de données et un calcul d'Itemsets candidats différents sont proposés par [ZAK 97]. Ces algorithmes supposent tous que la base de données est en format vertical (tid-lists pour chaque item) contrairement au partitionnement horizontal (des listes de transaction). La base de données est divisée entre les sites de façon à ce que chaque site obtienne un tid-list entière pour un seul item. En employant cette organisation verticale, le Support d'un Itemset peut être calculé simplement par l'intersection des tid-lists des items de cet Itemset. Cependant, ces algorithmes nécessitent une transformation de la base de données en format vertical si la base de données est organisée horizontalement. Ces algorithmes sont similaires et se diffèrent seulement dans la stratégie de recherche et dans la technique de décomposition en classes d'équivalence. Ces algorithmes nécessitent les 2-Itemsets fréquents pour partitionner les candidats, ils emploient une étape d'initialisation pour réunir les occurrences de tous les 2-Itemsets. Chacun de ces algorithmes a trois phases principales :

- La phase d'initialisation, qui effectue la distribution des données et des calculs ;
- La phase asynchrone, où chaque site génère indépendamment les Itemsets fréquents ;
- La phase de réduction, qui agrège les résultats finals.

Les comparaisons avec une implémentation de CD ont montré les ordres d'améliorations de Par-Max/Clique par rapport à CD.

- **Discussions :**

L'algorithme CaD est une sorte d'hybridation de CD et DD. L'algorithme se base sur la distribution des candidats de façon à assurer une cohérence entre les Itemsets qu'on calcule et les transactions qu'on utilise pour effectuer ces calculs. Bien qu'il emploie l'information spécifique du problème, CaD est moins performant que CD, à cause du coût de la redistribution nécessaire des Itemsets et des données entre chaque itération en parcourant la partition de la base de données locale à plusieurs reprises. HPA et HPA-ELD sont similaires à CaD. HPA emploie une technique de hachage pour adresser le déplacement des partitions de la base de données en déplaçant seulement les transactions (précisément les sous-ensembles de transactions) appropriées au site de destination.

Du fait que les candidats sont divisés par une fonction de hachage, les sous-ensembles des transactions sont aussi stockés par la même fonction de hachage. Donc le coût de communication total est réduit à O(p). HPA-ELD adresse le problème d'équilibrage de charge entre les sites par la réplication de quelques Itemsets extrêmement fréquents sur tous les sites. L'algorithme HD combine les avantages de CD et IDD, il réduit les coûts de communication des données par 1/G (G est le nombre de groupe). Il détermine le nombre de partitions de telle sorte que les partitions des candidats tiennent en mémoire principale de chaque site et chaque site ait assez de candidats au calcul. Il exploite aussi l'avantage de CD en échangeant juste l'information de Supports et en déplaçant le nombre minimal de transactions à travers le plus petit sous-ensemble de sites. HD semble avoir les mêmes performances que CD, mais il peut manipuler une plus grande base de données. Dans SH, les candidats sont générés indépendamment par chaque site instantanément en parcourant la partition de la base de données. Ceci pose le problème de la charge de calcul pour générer les candidats (il doit vérifier si tous les k-1 sous-ensembles des k-Itemsets sont fréquents ou non dans chaque transaction tandis qu'Apriori vérifie seulement si l'Itemset est la jointure de deux L_{k-1}). De plus, L'algorithme SH essaye de résoudre l'insuffisance de la mémoire principale par le bobinage (spooling) des candidats au disque (appelé les *RUN*s dans SH). Mais, Un problème possible avec SH est qu'il peut y avoir trop de *RUN*s qui présentent beaucoup d'entrée-sortie sur le disque.

Tous ces algorithmes utilisent le format horizontal de données et seul les algorithmes PAR-ECLAT et PAR-MaxEclat basés sur le concept de classe équivalence où chaque classe correspond à un sous ensemble d'arbre dans l'espace de recherche des Itemsets et qui peut être traité de façon asynchrone sur chaque site en utilisant le format vertical. PAR-ECLAT surpasse DD, CD, et CaD par plus d'un ordre, mais il ne calcule que les Itemsets fréquents maximums.

3.4 Synthèse :

Plusieurs algorithmes distribués pour l'extraction des fréquents ont été proposés ces dernières années pour le passage à l'échelle des grandes bases de données et de la puissance de calcul nécessaire. La plupart des ces algorithmes sont des parallélisations basées sur l'algorithme Apriori ou sur des améliorations de cet algorithme. Tous ces algorithmes adoptent des paradigmes de parallélisations spécifiques, les principaux paradigmes sont la parallélisation de données et la parallélisation des taches. Ces paradigmes sont appropriés à certaines situations et ils ont tous les deux des avantages et des inconvénients. Les algorithmes de type CD (le déplacement des résultats) considèrent le cas où les données sont partitionnées statiquement de façon homogène sur les différents sites tandis que l'ensemble des candidats est répliqué sur chaque site. Dans cette approche les communications entre les sites sont réduites au minimum du fait que seuls les Supports sont échangés par itération au détriment des calculs redondants en parallèle. Par contre, Les algorithmes de type DD (le déplacement des données) concentrent sur la maximisation du parallélisme ; ils diffèrent de CD dans le sens où, les sites se partagent le travail non plus par portion de base de données mais par candidats. Chaque site traite un sous ensemble de candidats disjoins, ce qui impose l'échange

des portions de la base de transaction des autres sites à chaque itération. Cette approche utilise la totalité de la mémoire du système, mais souffre de la grande charge de communication ce qui la rend peu pratique dans les contextes distribuées. Le schéma de communication en anneau et au lieu du broadcast peut réduire cette charge mais reste toujours un obstacle surtout dans le cas des grandes bases de données. Les algorithmes de type CaD (le déplacement des motifs) exploitent le domaine d'extraction dans le sens où il distribue les transactions et les candidats à base de leur préfixe de sorte que chaque site peut procéder indépendamment des autres. Cette technique de parallélisation est intéressante et n'entraînes pas d'autres communications/synchronisations, mais cette approche et selon le schéma résultant de la distribution des candidats peut souffrir d'un mauvais équilibre de charge. Les expérimentations ont démontré que les algorithmes de type CD font preuve d'une scalabilité optimale et d'une accélération excellente et surpassent les autres stratégies et que les algorithmes de type DD sont les mauvais tandis que ceux de type CaD présentent de bonnes performances au détriment de la charge élevée causée par le besoin de redistribution. Cependant, une approche prometteuse est de combiner les deux paradigmes, cette stratégie hybride a souvent les mêmes performances que le parallélisme de données mais elle peut traiter de grandes quantités de données, elle est plus scalable !

4 Contribution

L'étude faite sur les algorithmes d'extraction des Itemsets fréquents distribués a montré que ces derniers sont vus comme une alternative prometteuse et une évolution naturelle pour le problème d'extraction des règles d'associations.

Notre principale contribution dans ce chapitre vise à introduire des versions parallèles/distribuées d'algorithmes d'extraction d'Itemsets fréquents basés essentiellement sur les versions séquentielles de l'algorithme AprioriCM présenté dans le chapitre précédent.

Notre objectif consiste à introduire des algorithmes qui réduisent la quantité de messages échangés et le nombre de candidats générés en tirant profit de la notion de ***Support***$^{+}$ introduite précédemment.

Nous avons introduit deux versions d'algorithmes distribués AprioriDCM (Distributed AprioriCM) pour l'extraction distribuée des Itemsets fréquents dans les contextes distribués à savoir : AprioriDCM-Version avec validation par itération et AprioriDCM2-Version avec validation à postériori.

Nous présentons par la suite le schéma général de ces algorithmes.

4.1 Le Schéma de communication

Dans notre algorithme distribué AprioriDCM, nous avons adopté un schéma de communication en Maître/Esclaves. Nous distinguons ainsi deux types de sites : un site dit **Maître** et plusieurs sites dites **Esclaves**.

Le site **Maître** est responsable de la coordination d'un ensemble de sites dit Esclaves et de la consolidation des résultats de traitement issus des différents sites Esclaves.

Les sites **Esclaves** effectuent les traitements locaux sur les portions de données locales c'est-à-dire, la base de données est fragmentée horizontalement entre les sites Esclaves et chaque site Esclave est responsable sur les traitements concernant la portion de données dont il est responsable.

Le choix de ce type de schéma est justifié et plusieurs avantages en découlent de ce choix :

- **Réduire le coût de communication inter-sites:**

L'utilisation du schéma Maître/Esclaves permet de réduire considérablement le nombre de communications/synchronisations nécessaire pour le calcul distribué des Itemsets fréquents globaux par rapport à l'utilisation des schémas de communications usuels classiques des algorithmes parallèles/distribués présentés auparavant(le broadcaste total, l'élection de site, etc). Nous avons donc opté pour ce type de schéma de communication bidirectionnelle (entre chaque site esclave et le site maître) pour réduire le coût de communication entre les sites, nous allons illustrer cela par l'exemple suivant :

On suppose que la base de données est fragmentée horizontalement entre *P* sites. Soit C_k le nombre d'Itemsets candidats échangés pour l'itération k. C_k nous donne une idée sur la taille des messages échangés.

Nous avons donc estimé le coût des communications pour les cas suivants :

- **Le cas du broadcast** : Les algorithmes qui suivent ce schéma de communication exige à chaque itération k que chaque site *Pi* diffuse les Supports locaux calculés dans le site *Pi* à tous les autres sites ainsi, le coût de communication estimé pour ces algorithmes est :

$$Coût = \frac{1}{2}\sum_{k}[(p-1)\times p]\times C_k$$

- **Le cas de Maître/Esclave** : l'utilisation d'un site maître permet de réduire le nombre de messages échangés entre les sites puisque l'envoi des messages se fait directement vers un seul site Maître. A chaque itération k, les *P* sites esclaves envoient un message vers le site maître et ce dernier répond par un message vers tous les *P* sites esclaves. Ainsi, le coût de communication estimé pour un site maître et *P* site esclaves est :

$$Coût = \sum_{k} p \times C_k$$

Nous remarquons bien que le coût de communication dans le cas du broadcaste est plus élevé par rapport au schéma maître/esclaves. Le coût de communication dans le cas de broadcast est de l'ordre de $O(p^2)$ par rapport à $O(2p)$ dans le cas du Maître/esclaves.

- **Elimination des calculs redondants de la phase de génération des candidats:**

Certains algorithmes distribués d'extraction des Itemsets fréquents utilisant le broadcast pour l'échange de messages souffrent d'un inconvénient majeur : le calcul redondant des candidats au niveau de chaque site. A chaque itération, le même ensemble d'Itemsets fréquents est trouvé et le même calcul redondant de leurs Supports globales est effecutés dans chaque dans chaque site. Ce constat fait que ces algorithmes n'exploitent pas de manière efficace les ressources globales du système distribué.

Nous avons opté dans notre approche à la centralisation de la phase de génération des Itemsets candidats au niveau du site maître. Cela nous permet d'éviter les calculs redondants des Itemsets candidats dans chaque site esclaves et d'exploiter au mieux les ressources globales du système. Le site maître sera donc le seul responsable de la génération des Itemsets candidats et de leur diffusion à l'ensemble des sites esclaves.

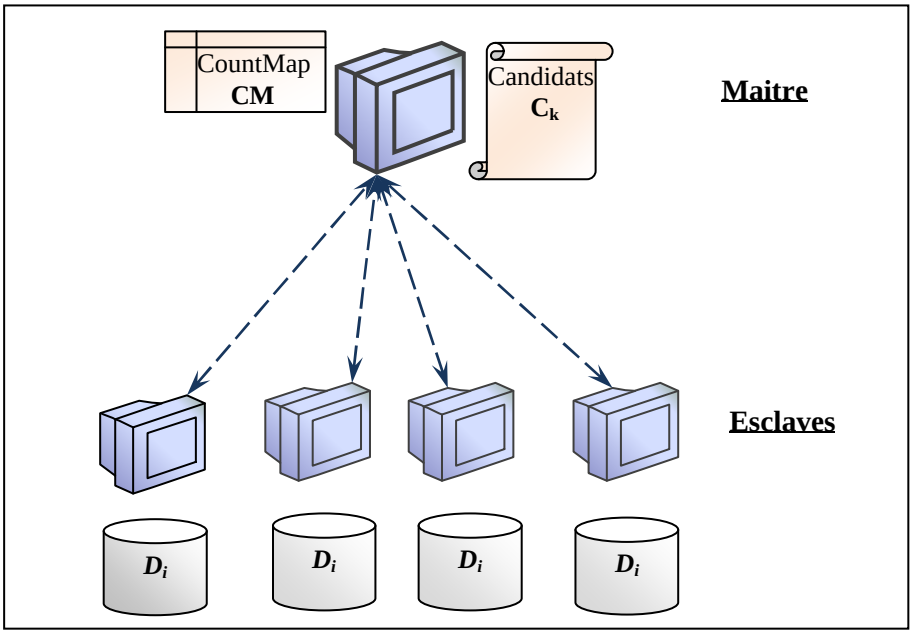


Fig. 2.3 Schéma de communication Maître/esclave

4.2 AprioriDCM : Version avec validation par itération

L'algorithme AprioriDCM proposé pour l'extraction distribuée des Itemsets fréquents est une parallélisation de la version de validation par itération de l'algorithme AprioriCM que nous avons adapté au contexte distribué expliqué précédemment. L'algorithme est composé de deux étapes :

- **Etape 1 : Construction de la structure CountMap globale**

Chaque site Esclave calcule sa structure CountMap local (LCountMap) en parcourant sa portion de base de données locale. Par la suite, le site Esclave envoie le contenu de LCountMap vers le site Maître.

- Le site Maître calcule la structure CountMap global (GCountMap) par la fusion (la somme) de tous les LCountMap reçus de différents Esclaves.
- Le site Maître déduit la liste des 1-Itemsets et 2-Itemsets fréquents globaux à partir de GCountMap et cela sans accès aux bases de données et donc sans communication avec les sites Esclaves.

➢ **Etape 2 : Extraction des k-Itemsets**

Le site Maître génère itérativement la liste des les k-Itemsets (k≥3). À chaque itération k:

- Le Maître construit la liste des k-Itemsets candidats par auto jointure des k-1- Itemsets générés dans l'itération précédente.
- Le site Maître calcule pour chaque candidat le Support$^+$, si ce Support$^+$ est inférieur à Minsup alors ce candidat est écarté.

  On rappelle que les Supports$^+$ sont calculés comme suite :

$$Support^+(Y) = \min_{|x|=|y|-1} Support\ (x)$$

- Le site Maître envoie la liste des k-Itemsets candidats à tous les sites Esclaves.
- Les sites Esclaves calculent les Supports locaux des k-Itemsets candidats reçus en parcourant leurs portions de base de données locales et ils renvoient les résultats au site Maître.
- Le site Maître détermine les k-Itemsets fréquents globaux en éliminant ceux qui ne satisfont pas le seuil minimal fixé de Support (*Minsup*).

L'étape deux est répétée jusqu'à ce qu'aucun candidat ne peut être généré par le maître.

```
1.  LCM(Si)= Gen-CountMap(D(i) ,I) ;          // Construction du CountMap local(LCM) dans chaque site Si
2.  Send(S(i),M, LCM(Si))                     // envoi du LCM(Si) au site Maitre M
3.  GCM=∑ LCM(Si)                             // Construction du CountMap Globale
4.  GL1 =GetFrequentItemsets(GCM,1) ;         // calcul des 1-Itemsets et 2-Itemsets fréquents Global
5.  GL2 =GetFrequentItemsets(GCM,2) ;
6.  Pour (k= 3; GLk-1≠Ø; k++) faire
7.  | GL_k^+ = CM-gen(GLk-1);                 // la phase de génération-élagage de candidats a base de Support+
8.  | Send(M,S(i), GL_k^+)                    // et leur diffusion aux sites Esclaves.
9.  |  pour chaque transaction t ∈ D(i) faire
10. |  |  Ct(i)= sous-ensemble (GL_k^+ ,t);          // calcul des supporst exacts
11. |  |  pour chaque candidates c ∈ Ct(i) faire     //  des Itemsets candidats
12. |  |  |  c.freq++;                               //  au niveau de chaque site Esclaves Si
13. |  |  fin;
14. |  fin;
15. |  Send(S(i),M, GL_k^+)
16. |  GLk={c∈ GL_k^+ |c.sup ≥ minsup}      // le site Maitre détermine les k-Itemset fréquenst globale
17. Fin
18. Résultats= ∪k GLk
```

Fig 2.4 Algorithme AprioriDCM

DATABASE *B*

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

SITE ESCLAVE 1

Tid	Items
1	ACD
2	BCE
3	ABCE

Site Maitre	
I	**A B C D E**
Minsup	2/6

SITE ESCLAVE 2

Tid	Items
4	BE
5	ABCE
6	BCE

Etape 1 : Construction de la structure CountMap globale

LCOUNTMAP1

A	2	1	2	1	1
B	2	2	0	2	
C	3	1	2		
D	1	0			
E	2				

GCOUNTMAP

A	3	2	3	1	2
B	5	4	0	5	
C	5	1	4		
D	1	0			
E	5				

LCOUNTMAP2

A	1	1	1	0	1
B	3	2	0	3	
C	2	0	2		
D	0	0			
E	3				

F_1	Support
A	3/6
B	5/6
C	5/6
E	5/6

F_2	Support
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

Etape 2 : Extraction des k-Itemsets

c_3	Support1
ABC	1
ABE	1
ACE	1
BCE	2

c_3	$Support^+$
ABC	min(2/6,3/6,4/6)=**2/6**
ABE	min(2/6,2/6,5/6)=**2/6**
ACE	min(3/6,2/6,4/6)=**2/6**
BCE	min(4/6,5/6,4/6)=**4/6**

c_3	Support2
ABC	1
ABE	1
ACE	1
BCE	2

F_3	Support
ABC	**2/6**
ABE	**2/6**
ACE	**2/6**
BCE	**4/6**

C_4	$Support^+$
ABCE	min(2/6,2/6,2/6,4/6)=**2/6**

C_4	Support1
ABCE	1

c_3	Support2
ABCE	1

F_4	Support
ABCE	**2/6**

Notre algorithme nécessite au total trois parcours de la base de données et donc trois échanges entre les sites pour le calcul de l'ensemble des Itemsets fréquents de l'exemple précédent.

Nous reprenons l'exemple précédent pour le cas de l'algorithme *AprioriDistribué* avec un site maître et deux sites esclaves. L'algorithme *AprioriDistribué* une parallélisassions selon le schéma Maître/esclaves de l'algorithme Apriori présenté dans le chapitre précédent.

- A la première itération, les deux sites esclaves calculent les Supports locaux des 1-Itemests candidats ($C_1^{(P1)}$={A,B,C,D,E}, $C_1^{(P2)}$={A,B,C,D,E}) et envoient ces calculs au site maître.
- Le site maitre fusionne les candidats reçus ($C1^{(P1)}$ et $C1^{(P2)}$), détermine les 1-Itemsets fréquents F_1, calcul les candidats C_2 et les renvoient aux sites esclaves P1 et P2.
- Ce processus est répété pour l'itération 2, 3 et 4 avec respectivement les candidats C_2={AB,AC,AE,BC,BE,CE}, C_3={ABC,ABE,ACE,BCE} et C_4={ABCE}.

L'algorithme *AprioriDistribué* performe donc quatre itérations pour le calcul des Itemsets fréquents et par conséquent quatre phases de communications entre le site maitre et les sites eslaves.

En conséquence et selon cet exemple illustratif, l'algorithme AprioriDCM nécessite trois itérations et donc trois phase de communication au lieu de quatre pour le cas d'AprioriDistribué.

$$Cout_{AprioriDCM} < Cout_{AprioriD}$$

4.2.1 Evaluation de AprioriDCM- Version avec validation par itération

Nous avons effectué des expérimentations pour évaluer les performances de notre algorithme AprioriDCM par rapport à l'algorithme classique AprioriDistribué (une version distribuée d'Apriori que nous avons adapté au schéma Maître/Esclaves).

Les algorithmes sont implémentés en Java et des expérimentations sont effectuées dans un réseau local. Les bases de données sont fragmentées horizontalement et distribuées sur les sites Esclaves.

Nous avons effectué des tests sur un réseau local composé d'un site Maître et d'un nombre de sites Esclaves respectivement de trois (03), six (06) et neuf (09) sites Esclaves.

Les figures suivantes représentent les résultats réalisés par l'algorithme AprioriDCM et AprioriDistribué pour les bases de données BDDT3333I16 et BDDT1000kI16:

Résultats pour La base de données BDDT3333I16						
MinSup (%)	Temps d'exécution de AprioriD (s)			Temps d'exécution de AprioriDCM (s)		
	3 Esclaves	6 Esclaves	9 Esclaves	3 Esclaves	6 Esclaves	9 Esclaves
50	0.028	0.041	0.053	0.027	0.037	0.051
40	0.071	0.079	0.103	0.044	0.055	0.073
30	0.170	0.201	0.305	0.115	0.128	0.276
20	0.230	0.311	0.396	0.194	0.268	0.309

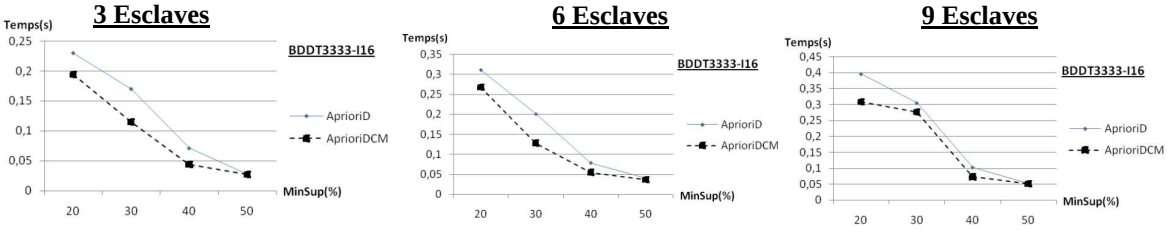


Fig. 2.5 Temps d'exécution : AprioriDCM vs AprioriDistribué avec BDDT3333I16

Les résultats réalisés avec la base de données BDDT3333I16 qui contient un nombre réduit de transactions ont montré que l'algorithme AprioriDCM surpasse l'algorithme AprioiriDistribué surtout pour des valeurs de Supports faibles.

Cela est dû au nombre réduit d'Itemsets candidats générés par l'algorithme AprioriDCM par rapport à l'algorithme AprioriDistribué ce qui influe directement sur le coût de communication où le volume des messages échangé entre les sites est réduit par rapport à l'algorithme AprioriDistribué. De plus, l'algorithme AprioriDCM nécessite au minimum une itération de moins que l'algorithme AprioriDistribué et donc une phase de communication de moins.

Nous avons augmenté le nombre de transactions de la base de données pour tester la scalabilité de notre algorithme. Les figures suivantes illustrent les résultats réalisés par les deux algorithmes AprioriDCM et AprioriDistribué :

Résultats pour La base de données BDDT1000KI16						
MinSup (%)	Temps d'exécution de AprioriD (s)			Temps d'exécution de AprioriDCM (s)		
	3 Esclaves	6 Esclaves	9 Esclaves	3 Esclaves	6 Esclaves	9 Esclaves
50	1.103	1.068	0.978	0.297	0.340	0.863
40	4.007	3.123	3.439	0.992	0.800	1.332
30	15.118	11.896	13.120	10.853	8.801	8.668
20	26.089	25.071	24.755	22.979	21.057	20.720

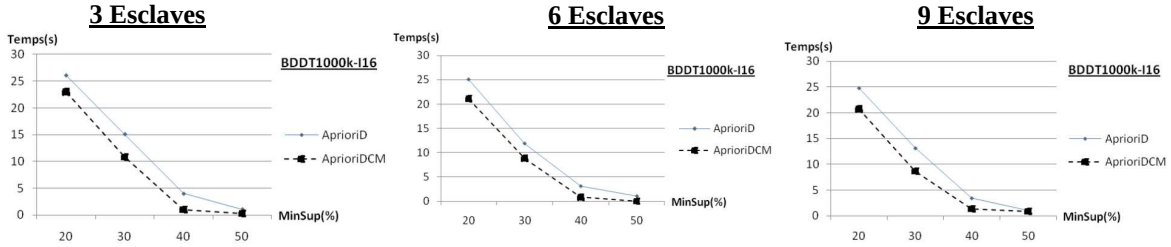


Fig. 2.6 Temps d'exécution : AprioriDCM vs AprioriDistribué avec BDDT1000kI16

Les résultats ont montré que l'algorithme AprioriDCM garde sa supériorité sur la version distribuée de l'algorithme classique AprioriDistribué. L'algorithme AprioriDCM enregistre une bonne scalabilité par rapport à la taille des bases de données (nombre de transactions).

4.3 AprioriDCM2 : Version avec validation à postériori

L'algorithme AprioriDCM2 est la version parallèle de l'algorithme AprioriCM2 (version avec validation à posteriori) présenté auparavant pour l'extraction distribuée des Itemsets fréquents.

Nous rappelons qu'une réflexion est faite sur cette version qui nécessite que deux parcours de la base de données. Dans cette optique, la parallélisation de l'algorithme AprioriCM2 n'auras besoins que de deux phase de communications entre les sites, de plus la phase de génération des candidats (phase 2) sera exécuté indépendamment sur le site Maitre et sans aucune communications avec les sites esclaves.

L'algorithme AprioriDCM2 est composé de trois étapes :

- **Etape 1 : Construction de la structure CountMap globale**

Cette phase est identique à l'algorithme AprioriDCM précédente : chaque site esclave calcule sa structure CountMap locale (LCounMap) en parcourant leurs bases de données locales ; Le site maître construit la structure CountMap globale (GCountMap) en effectuant la somme des valeurs des différents LCountMap reçus des sites esclaves.

- **Etape 2 : Extraction des k-Itemsets fréquents globaux**

Le site maître construit itérativement la liste des k-Itemsets candidats globaux (k≥3) en utilisant le Support$^+$: Le site maître calcule pour chaque candidat le Support$^+$ global, si ce Support$^+$ global est inférieur à *Minsup* alors ce candidat est écarté :

Insert into Ck Select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$
From L_{k-1} p, L_{k-1} q
Where p.item$_1$=q.item$_1$, …, p.item$_{k-2}$=q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$
and Support$^+$(p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$) ≥ *Minsup*

On rappelle que cela est fait sans aucun accès ou parcours de la base de données. Le calcul de Support$^+$ des Itemsets est effectué de façon récursive comme suit :

$$Support^+(y) = \begin{cases} \min\limits_{|x|=|y|-1} Support(x), & |y| = 3 \\ \min\limits_{|x|=|y|-1} Support^+(x), & |y| > 3 \end{cases}$$

Cette étape est répétée jusqu'à ce qu'aucun candidat ne puisse être généré. De plus, durant cette phase, aucune communication avec les sites esclaves n'est nécessaire.

➢ **Etape 3 : raffinage des fréquents globaux :**

A ce stade d'algorithme, nous avons calculé les k-Itemsets fréquents globaux et comme nous avons signalé auparavant, une phase de validation doit être effectuée pour raffiner cet ensemble d'Itemsets :

- Le site maître envoie aux sites esclaves la liste des k-Itemsets candidats construite dans l'étape précédente.
- Les sites esclaves calculent les Supports locaux des k-Itemsets reçus en parcourant leurs portions locales de base de données et renvoient les résultats au site maître.
- Le site maître détermine les k-Itemsets fréquents globaux en éliminant les non globalement fréquents. On, signale ici que les ensembles des 1-Itemsets et 2-Itemsets ne sont pas inclus dans ce parcours du fait qu'on a calculé leurs Supports exacts à l'étape 1 de l'algorithme via la structure CountMap globale.

1. **LCM**$^{(Si)}$= *Gen-CountMap*($D^{(i)}$,I) ; *// Construction du CountMap local(LCM) dans chaque site S_i*
2. *Send($S^{(i)}$,M,* **LCM**$^{(Si)}$*)* *// envoi du LCM$^{(Si)}$ au site Maitre M*
3. GCM=$\sum$ **LCM**$^{(Si)}$ *// Construction du CountMap Globale*
4. GL_1 =*GetFrequentItemsets*(GCM,1) ; *// calcul des 1-Itemsets et 2-Itemsets fréquents Global*
5. GL_2 =*GetFrequentItemsets*(GCM,2) ;
6. GL_2^+ = GL_2 ;
7. **Pour** (k= 3; $GL_{k-1}^+ \neq \emptyset$; k++) **faire**
8. | GL_k^+ = *CM-gen*(GL_{k-1}^+); *// la phase de génération-élagage de candidats a base de Support$^+$*
9. **fin;**
10. CGL^+ ={$\cup$ GL_k^+ |k ≥ 3} ; *// la liste de tous les candidats k≥3*
11. *Send(M, $S^{(i)}$, CGL^+) ;* *// envois de la liste à chaque site Si*
12. **pour chaque** transaction t ∈ $D^{(i)}$ **faire**
13. | $C_t^{(i)}$= sous-ensemble (CGL^+ ,t); *// calcul des supporst exacts*
14. | **pour chaque** candidates c ∈ $C_t^{(i)}$ **faire** *// des Itemsets candidats*
15. | | c.freq++; *// au niveau de chaque site Esclaves Si*
16. | **fin**;
17. **fin;**
18. *Send($S^{(i)}$,M, CGL^+) ;*
19. GL_k={c∈ CGL^+ |c.sup ≥ minsup} *// le site Maitre détermine les k-Itemset fréquenst globale k≥3*
19. **Résultats=** $\cup_k$ **GL_k**

Fig 2.7 Algorithme AprioriDCM

DATABASE *D*

Tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

SITE ESCLAVE 1

Tid	Items
1	ACD
2	BCE
3	ABCE

Site Maitre	
I	**A B C D E**
Minsup	2/6

SITE ESCLAVE 2

Tid	Items
4	BE
5	ABCE
6	BCE

Etape 1 : Construction de la structure CountMap globale

LCOUNTMAP1

A	2	1	2	1	1
B	2	2	0	2	
C	3	1	2		
D	1	0			
E	2				

GCOUNTMAP

A	3	2	3	1	2
B	5	4	0	5	
C	5	1	4		
D	1	0			
E	5				

LCOUNTMAP2

A	1	1	1	0	1
B	3	2	0	3	
C	2	0	2		
D	0	0			
E	3				

F_1	Support
A	3/6
B	5/6
C	5/6
E	5/6

F_2	Support
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

Etape 2 : Extraction des k-Itemsets globaux

c_3	$Support^+$
ABC	min(2/6,3/6,4/6)=**2/6**
ABE	min(2/6,2/6,5/6)=**2/6**
ACE	min(3/6,2/6,4/6)=**2/6**
BCE	min(4/6,5/6,4/6)=**4/6**

C_4	$Support^+$
ABCE	min(2/6,2/6,2/6,4/6)=**2/6**

Etape 3 : raffinage des Itemsets fréquents globaux

C_k	Support1
ABC	1
ABE	1
ACE	1
BCE	2
ABCE	1

C_k	Support1
ABC	1
ABE	1
ACE	1
BCE	2
ABCE	1

F_k	Support
ABC	**2/6**
ABE	**2/6**
ACE	**2/6**
BCE	**4/6**
ABCE	**2/6**

L'algorithme AprioriDCM2 nécessite au total deux parcours de la base de données (un parcours dans l'étape1 et un autre dans l'étape 3).

L'étape2 de l'algorithme est exécutée au niveau du site Maître et aucun échange n'est effectué avec les sites esclaves. Ainsi, l'algorithme nécessite donc deux échanges entre les sites au lieu de quatre dans le cas d'AprioriDistribué.

4.3.1 Evaluation d'AprioriDCM2- Version avec validation à postériori

De même, nous avons effectué des expérimentations pour évaluer les performances d'AprioriDCM2 par rapport à l'algorithme classique AprioriDistribué avec les bases de données précédentes et les mêmes conditions d'expérimentations.

Les figures suivantes représentent les résultats obtenus:

Résultats pour La base de données BDDT3333I16						
MinSup (%)	Temps d'exécution de AprioriD (s)			Temps d'exécution de AprioriDCM2 (s)		
	3 Esclaves	6 Esclaves	9 Esclaves	3 Esclaves	6 Esclaves	9 Esclaves
50	0.028	0.041	0.053	0.056	0.088	0.094
40	0.071	0.079	0.103	0.058	0.077	0.099
30	0.170	0.201	0.305	0.102	0.139	0.161
20	0.230	0.311	0.396	0.216	0.254	0.359

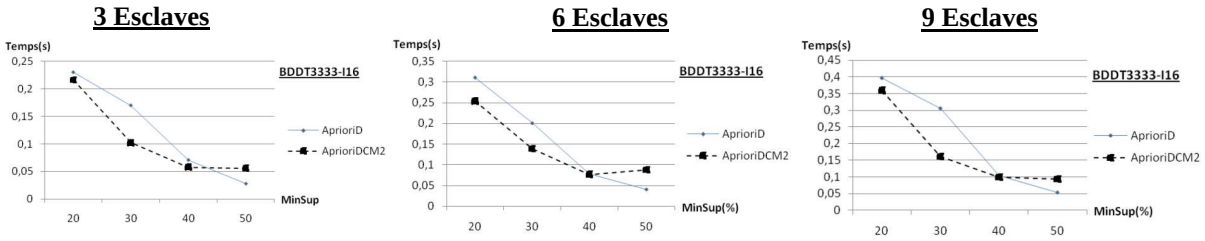


Fig. 2.8 Temps d'exécution : AprioriDCM2 vs AprioriDistribué avec BDDT3333I16

Les résultats montrent que l'algorithme AprioriDCM2 surpasse l'algorithme AprioriDistribué dans ce contexte distribuée et cela à l'inverse des résultats trouvé dans le contexte séquentiel où la version AprioriCM2 est moins performante que l'algorithme Apriori à cause du nombre d'Itemsets candidats supplémentaire générés par AprioriCM2 et le cout de la phase de raffinage des résultats.

Or, dans le contexte distribué où les données et les calculs sont distribués nous constatons une meilleure performance de l'algorithme AprioriDCM2 par rapport à Apriori. Cette amélioration est en effet à cause du nombre réduit de phases de communications nécessaire pour l'algorithme AprioriDCM2 pour le calcul de l'ensemble des Itemsets globalement fréquents. Nous remarquons que le coût réduit des communications a pu couvrir l'handicape du nombre supplémentaire d'Itemsets candidats généré par l'algorithme AprioriDCM2 (nous avons un recouvrement du temps de communication par la puissance des calculs.).

Par la suite, nous avons augmenté d'avantage la taille de la base de données pour tester la scalabilité de l'Algorithme AprioiDCM2, la figure suivante montre les résultats obtenus :

Résultats pour La base de données BDDT1000KI16						
MinSup (%)	Temps d'exécution de AprioriD (s)			Temps d'exécution de AprioriDCM2 (s)		
	3 Esclaves	6 Esclaves	9 Esclaves	3 Esclaves	6 Esclaves	9 Esclaves
50	1.103	1.068	0.978	2.157	1.741	1.913
40	4.007	3.123	3.439	2.169	1.983	2.107
30	15.118	11.896	13.120	6.770	7.525	7.763
20	26.089	25.071	24.755	23.093	22.850	24.463

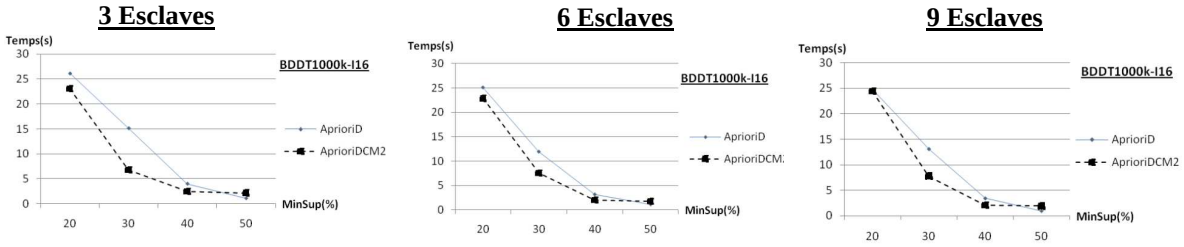


Fig. 2.9 Temps d'exécution : AprioriDCM2 vs AprioriDistribué avec BDDT1000kI16

Les résultats obtenus ont montré que l'algorithme AprioriDCM2 garde sa supériorité sur l'algorithme AprioriDistribué.

Nous avons remarqué d'ailleurs que les résultats réalisés par les deux algorithmes AprioriD et AprioiriDCM2 ont tendance à se rapprocher dans le cas où le nombre de sites (nœuds Esclaves) est élevé (voir le cas d'utilisations de 09 neouds Esclaves).

Cela est dû essentiellement à la granularité trop fine de la distribution des données. Nous rappelons que la recherche de règles d'associations est basée sur des critères globaux et donc une granularité trop fine dans la distribution des données engendra des coûts de communications coûteuses pour le calcul des motifs globaux. Par conséquent, l'algorithme AprioriDCM2 perdra l'avantage du recouvrement du coût de communication par la puissance de calcul.

4.4 Synthèse des résultats d'AprioriDCM

Les expérimentations réalisés nous ont permet d'évaluer notre algorithme distribué avec ses deux versions.

Les résultats montrent que l'algorithme AprioriDCM (Version avec validation par itération) est plus performant et plus scalable que l'algorithme AprioriDistribué. L'algorithme enregistre une bonne accélération avec l'augmentation de la taille de la base de données et avec l'augmentation du nombre de nœuds et donc du taux de parallélisme.

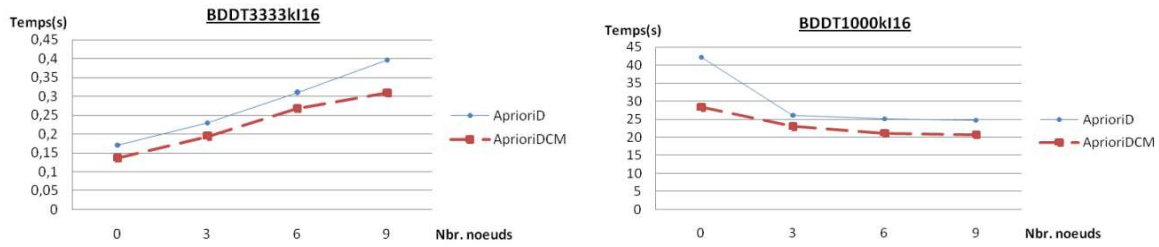


Fig. 2.10 Scalabilité de AprioriDCM par nombre de noeuds

L'algorithme AprioriDCM réalise une meilleure accélération que l'algorithme AprioiriDistribue. Plus le nombre de nœuds est élevé (augmentation du taux de parallélisme) plus les performances sont meilleurs. En outre, plus le nombre de transactions à traiter est grand plus l'accélération est meilleur.

Pour le cas de la version AprioriDCM2 (version avec validation à posteriori), les résultats montre que cette version aussi est plus performante que l'algorithme AprioriDistribué et elle enregistre une bonne accélération avec l'augmentation du nombre de nœuds (le taux de parallélisme) et avec l'augmentation du nombre de transactions.

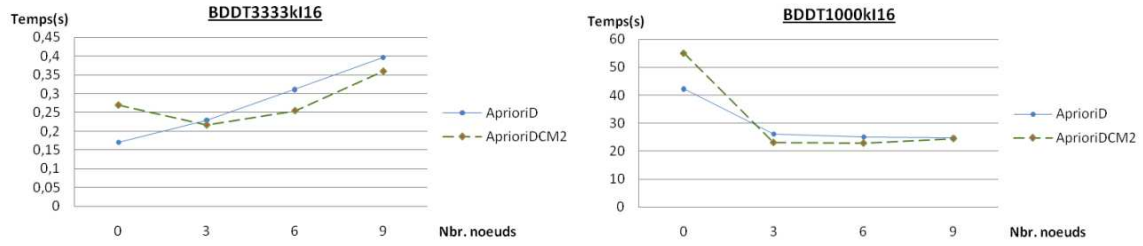


Fig. 2.11 Scalabilité de AprioriDCM2 par nombre de noeuds

Les expérimentations ont montré l'utilité de la réflexion faite sur cette version pour remédier à la lenteur de la phase de communication et cela par la limitation du nombre des communications et la tolérance du certains nombre supplémentaire d'Itemsets candidats et donc le recouvrement du cout de communication par la puissance de calcul.

Or, nous constatons que cette version perdre cet avantage dans le cas d'une granularité trop fine du parallélisme (nombre de nœuds) et ce rapproche des performances de l'algorithme AprioriDistribué.

5 Conclusion :

La réalisation d'une bonne performance sur les systèmes distribués d'aujourd'hui n'est pas triviale. Les défis principaux incluent la minimisation de synchronisation et de communication, l'équilibre de charge de travail, la bonne disposition des données, la décomposition de ces données et la minimisation des entrées-sorties disques.

L'algorithme distribué AprioriDCM a été conçu pour répondre aux spécificités des Supports d'exécution de type réseau de stations de travail ou Grilles de calculs à savoir la favorisation des traitements parallèles, la réduction du coût de communications et/ou de synchronisations. La version AprioriDCM2 favorise les traitements indépendants entres les noeuds distribués, cependant, cette version soufre du problème des Itemsets parasites ce qui nécessite une phase de raffinage des résultats. Or, l'algorithme permet de réduire le nombre de parcours de la base de données nécessaire à deux parcours seulement.

La version AprioriDCM permet de contourner le problème des parasites en admettant des communications à la fin de chaque itération pour raffiner l'ensemble d'Itemsets. Les expérimentations montrent la supériorité de AprioriDCM par rapport à la version AprioriDCM2 et confirme le rôle d'utilisation du nœud maître pour la validation des résultats à chaque itération plutôt qu'à posteriori.

1 Introduction :

Ce chapitre présente certaines solutions visant à exploiter les Grilles pour le Support du DataMining distribué et en particulier la technique des règles d'associations. Les Grilles sont considérées comme des plateformes décentralisées et haute performance où il est possible d'exécuter les tâches de DataMining et les applications de découverte des connaissances.

Plusieurs projets de recherches traitent le sujet de DataMining sur les plateformes de Grilles qui visent la mise en œuvre des plateformes Grilles bien adaptées aux problèmes de DataMining distribués à grande échelle. Ce mariage entre les techniques de DataMining et les environnements Grilles ouvre plusieurs axes de recherches qui couvrent plusieurs aspects du problème du Support de DataMining distribué sur les plateformes Grilles et ils se résument dans les points suivants:

- L'adaptation de l'architecture des environnements de Grilles pour le DataMining ;
- Le choix d'approche appropriée pour le DataMining distribué : dans certains scénarios les données qui pour une raison ou une autre ne peuvent pas être intégrées dans un seul nœud de calcul.
- L'identification des ressources pour l'exécution des tâches de DataMining à grande échelle (sources de données, les programmes, les capacités de calcul et de stockage).
- Le problème de la confidentialité et la sécurité des données.
- Les outils qui facilitent le DataMining dans les environnements de calcul distribué : les outils de visualisation, les interfaces utilisateurs, les langages et les modèles de programmation.
- L'adaptation des algorithmes de DataMining existants et/ou le développement de nouveaux algorithmes qui peuvent exploiter les architectures de calculs distribués.

Ces activités de recherche se traduisent par le lancement de plusieurs projets de développement de systèmes Grilles pour le DataMining. Ces systèmes adoptent différentes approches, les projets de recherche comme le projet TeraGrid et GridDataMining visent à développer des services de DataMining sur les Grilles tandis que des systèmes comme le Knowledge Grid, Discovery Net, et GridMiner visent à construire des systèmes complets de KDD pour concevoir des processus distribués de découverte de connaissances sur les Grilles (Grille de connaissances).

Dans ce chapitre, nous introduisons premièrement le concept de Grille informatique et nous présentons, ensuite, les caractéristiques des principaux environnements de Grilles dédiés aux applications de DataMining distribué. Nous présentons après quelques tentatives et travaux de recherche d'implémentation de la technique d'extraction des règles d'associations dans ces environnements et/ou dans des environnements de Grilles de tests. A la fin de ce chapitre, nous abordons notre approche pour le problème d'extraction des règles d'associations en présentant l'adaptation de notre méthode distribuée pour une utilisation sur un environnement Grille.

2 Présentation des Grilles

Les Grilles ont émergé ces dernières années comme un important domaine de calcul distribué qui risque bien de révolutionner à de nombreux égards la majorité des domaines de recherches scientifiques voire même la façon dont nous abordons l'informatique actuellement.

Durant les fins des années 90, les Grilles ont commencé comme des projets qui relient des sites de calcul pour fournir des ressources d'un grand rang d'applications de hautes performances. Le but principal d'une Grille est de mutualiser des ressources informatiques (puissance de calcul et espace de stockage) à l'échelle mondiale en créant pour l'utilisateur l'illusion d'un grand et puissant ordinateur virtuel qui serait la connexion d'un très grand nombre de systèmes hétérogènes reliés par le réseau.

Le Project I-WAY [FOS 95] est généralement considéré comme la première Grille moderne. Le projet a commencé en 1995 comme un réseau expérimental haute performance reliant plusieurs ordinateurs et des environnements de visualisations avancées. L'idée était d'unifier les ressources dans les centres de calcul (Environ de 17 sites ont été reliés ensemble par réseau et plus de 60 applications ont été développées et déployées comme une infrastructure Grille).

Le terme Grid a été choisi par analogie au réseau de distribution d'électricité (appelé Electrical Power Grid en Anglais) [FOS 01]. En effet, on peut rapprocher la puissance de calcul actuelle avec ce qu'était l'électricité au début du XXe siècle. On maîtrisait l'électricité et on disposait de matériel pour l'exploiter mais chaque personne devait produire sa propre électricité et la consommer sur place. Ce n'est qu'avec le développement du réseau de distribution électrique que la vraie révolution a pu s'opérer en offrant, au plus grand nombre, la possibilité de recevoir et utiliser de l'électricité sans se soucier de la façon ou de l'endroit où elle a été produite. L'idée des Grilles Informatiques envisage la même démarche en informatique : disposer d'une prise de raccordement de son ordinateur au réseau informatique, mais pas seulement pour y trouver des données comme sur Internet, mais pour y trouver de la puissance de calcul et de l'espace de stockage.

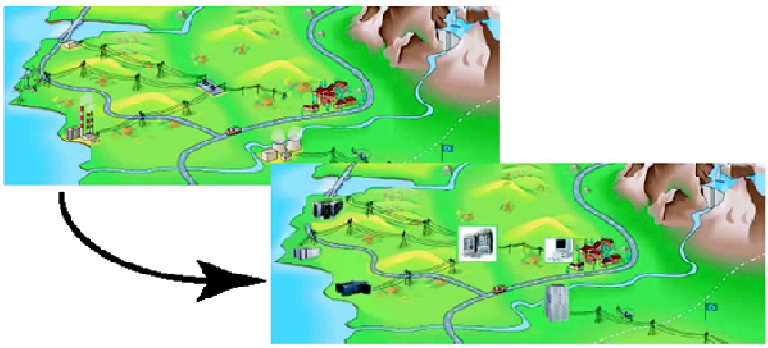


Fig. 3.1 Une Grille informatique, comme l'on a des grilles d'électricités

Par ailleurs, les Grilles ne sont pas seulement des infrastructures de calcul pour des applications parallèles et distribuées à grande échelle, mais aussi des technologies qui peuvent mettre en relation et unifier diverses ressources.

Les ressources peuvent être des systèmes de stockage, des ressources de données ou tout autre dispositif de calcul ou de communication géographiquement distribué et appartenant éventuellement à différentes organisations.

- **Définition :**

Le concept de Grille de calcul étant encore relativement récent, on peut en trouver de nombreuses définitions que ce soit dans la littérature scientifique ou sur Internet. L'une des définitions présentes le « Grid Computing » comme le successeur du « métacomputing ». Le « métacomputing » regroupe l'ensemble des techniques permettant d'utiliser plusieurs superordinateurs au sein d'une même application, en tenant compte des temps de communications sur des réseaux à longue distance lors de la conception de l'application. L'extension de cette technique a un grand nombre de superordinateurs, prenant en compte les problèmes de répartition et d'équilibrage de charge, et réalisant une tolérance aux pannes, constitue l'une des approches du Grid Computing.

Une autre définition du « Grid Computing» correspond à certains projets très médiatisés, comme SETI@home [SET], qui orchestre des centaines de milliers de PC. Il s'agit d'application massivement parallèle, où chaque nœud de la Grille exécute une version complète du programme avec certains paramètres ou un certain jeu de données. L'ensemble des exécutions de la Grille permet de réaliser plus rapidement la totalité des calculs.

Une troisième définition donnée dans [FOS 03] « Une Grille de calcul est une infrastructure matérielle et logicielle qui fournit un accès fiable, uniformisé, répandu et bon marché à des ressources informatiques de hautes performances ».

Nous parlons d'une **infrastructure** car une Grille de calcul vise à mutualiser un grand nombre de ressources (cycles CPU, données, espace de stockage). Pour ce faire, il est nécessaire de disposer d'une architecture matérielle permettant l'interconnexion de ces ressources, mais aussi d'une architecture logicielle afin de pouvoir faire cohabiter, surveiller et contrôler cet ensemble.

La nécessité d'un service **fiable** est fondamentale. Les utilisateurs doivent avoir l'assurance qu'ils auront accès à un service prévisible, constant et de haute performance de la part des différents composants qui constituent la Grille. Selon le type de programme exécuté sur la Grille, la signification que l'on donne à cette performance peut être de différentes natures telles que la bande passante du réseau, le temps de latence, la puissance de calcul brute, les services logiciels disponibles ou encore la sécurité.

Avoir un service **uniformisé** est le second objectif de la Grille. Comme pour l'électricité, il est important d'avoir des services standardisés accessibles via des interfaces standards. Sans

de telles normes, le développement d'applications ainsi que le déploiement de la Grille sont fortement compromis voire impossibles.

Un des grands défis des Grilles est d'arriver à concevoir de tels standards afin de gérer des matériels et des logiciels très hétérogènes tout en conservant la performance.

Un accès **répandu** garantit de pouvoir retrouver les différents services quel que soit l'environnement vers lequel on pourrait migrer. Cela ne signifie pas que les ressources sont accessibles de n'importe où universellement. De même qu'une nouvelle maison n'a accès à l'électricité qu'une fois le câblage installé et après avoir souscrit chez un fournisseur, l'accès à la Grille aura une circonscription et un contrôle d'accès similaire. On pourra toutefois compter sur un accès suffisamment universel pour pouvoir être utilisé quel que soit l'environnement d'exécution.

Enfin, l'accès à cette infrastructure doit être relativement **bon marché** au regard des bénéfices qu'elle peut apporter. Les industriels dépensent des dizaines de millions pour construire des centrales électriques car cet investissement leur est rentable. D'autre part, l'investissement financier pour pouvoir bénéficier du réseau électrique est, bien heureusement, infiniment plus restreint. Idéalement, la Grille de calcul devrait jouir des mêmes attraits économiques.

Finalement, nous citons la définition du ministère français de la recherche qui résume toutes ces définitions et définit la Grille comme la **G**lobalisation des **R**essources **I**nformatiques et des **D**onnées (GRID) [SAB 04].

2.1 L'évolution des Grilles :

Dans la fin des années 90, les chercheurs de Grilles se sont réunis ensemble dans le Grid Forum(GF), qui est étendu par la suite à Global Grid Forum(GGF) [GGF], où plusieurs recherches sont effectuées pour définir des standards pour les Grilles comme la spécification architectural OGSA qui intègre Globus et les approches des services Web.

- **Open Grid Service Architecture (OGSA):**

OGSA [FOS 02] est une spécification informative qui définit le plan directeur sur comment une Grille doit apparaître en incluant l'infrastructure. Elle définit aussi le modèle de programmation des services de la Grille. Elle donne des instructions sur comment créer des services de Grille en donnant les grandes lignes des composants nécessaires pour créer et délivrer une solution Grille.

- **Open Grid Service Infrastructure (OGSI):**

OGSI [TUE 03] est une spécification de l'infrastructure d'OGSA. C'est un middleware pour les Grilles services qui définit les grandes lignes sur les mécanismes de création, de gestion et d'échange d'informations pour les services de la Grille. OGSA adopte actuellement une nouvelle génération des technologies de Grilles qui est devenue un standard pour la création d'applications orientées services qui sont basées sur les Services Web.

- **Les Web Services [W3C 02]:**

Les protocoles traditionnels pour le calcul distribué comme les Sockets, les RPC, le Java RMI, et CORBA, etc. ne sont pas basés sur des standards et il est difficile de les utiliser dans des environnements hétérogènes. Il y a aujourd'hui les Services Web qui sont des standards ouverts. La technologie des services Web, en pleine expansion, offre une méthode simple, souple et portable d'échanges d'informations entre des applications à travers un réseau. Pour l'instant, les infrastructures des middlewares basés sur XML peuvent créer et intégrer des applications dans des environnements hétérogènes indépendamment des plateformes, les langages de programmations ou des locations. Cette technologie est Basée sur le protocole SOAP (Simple Object Access Protocol) [W3C 00] qui représente un protocole de communication pour les clients et les serveurs pour échanger des messages en format XML à travers un protocole HTTP. Les services web fournissent une plateforme prometteuse pour les systèmes Grilles.

- **Web Services Resource Framework (WSRF) [CZA 04]:**

OGSA est basé sur les services web et elle standardise quasiment les servies de la Grilles comme la gestion des jobs et des ressources, la communication et la sécurité. Le WSRF est un standard pour la modélisation ressources avec état avec les Web Services. Actuellement, OGSI est remplacé par le WSRF qui est un ensemble de spécifications de services web lancé en janvier 2004. Un des avantages des WSRF est qu'il partitionne les fonctionnalités d'OGSI en un ensemble de spécification et il travail bien avec les outils de services web par comparaison à OGSI. Le WSRF est supporté par Globus (GT4) [FOS 05].

2.2 Organisation virtuelle

Les ressources d'une Grille peuvent potentiellement être utilisées par une très large communauté d'utilisateurs. Il est donc nécessaire d'avoir une politique claire pour la gestion des droits associés aux ressources et à ceux qui peuvent les utiliser. Pour ce faire, on regroupe les utilisateurs en organisations virtuelles [FOS 01] (Virtual Organisation (VO)). Toutes les personnes appartenant à une même VO ont généralement des droits et besoins communs, typiquement parce qu'elles travaillent dans une même discipline. La durée de vie de ces organisations virtuelles peut être variable, ainsi que les buts qu'elles poursuivent. Ainsi, un utilisateur désireux de bénéficier des ressources de la Grille doit obtenir un certificat d'une autorité de confiance puis le faire enregistrer auprès d'un ou plusieurs VO. Ceci fait, il peut alors accéder aux ressources et infrastructures allouées aux organisations dont il est membre : processeurs, espace disque, données et résultats d'expériences.

2.3 Les middlewares Grilles

Le middleware est la brique de base regroupant l'ensemble des éléments logiciels pour la mise en œuvre d'une Grille. Il comprend notamment les fonctions suivantes :

- Le partage et l'allocation des différentes ressources de la Grille suivant des critères techniques de performance, mais également des critères économiques et d'éventuelles contraintes utilisateurs.
- L'exécution, l'ordonnancement et l'administration de la Grille, intégrant toutes les fonctions de monitoring et de gestion.
- L'ensemble des procédures de sécurisation de la Grille, notamment les outils d'authentification des utilisateurs, la gestion des restrictions accès, la confidentialité des données et des résultats.
- Les outils collaboratifs permettant aux divers acteurs de travailler ensemble et d'échanger les documents, les données, les logiciels, les résultats, etc., en garantissant la cohérence de ceux-ci au cours de l'ensemble des manipulations.
- Les outils d'évaluation des performances et de mesure de la qualité de service.
- Les outils de développement et les interfaces utilisateurs pour le déploiement des différentes applications.

Ces middlewares s'appuient sur des protocoles standards de l'Internet tels que FTP (File Transfer Protocol), LDAP (Light Directory Access Protocol), HTTP (HyperText Transfert Protocol). Parmi les middlewares les plus utilisés actuellement il faut citer les outils : GLOBUS [FOS 97 ;01 ;05], LEGION [CHA 99] et UNICORE [UNI 03].

- **Globus**

Globus fournit une infrastructure logicielle qui permet aux applications de voir les ressources distribuées comme une machine virtuelle unique. Le projet Globus [FOS 98 ; GLO] est un effort multi institutionnel Américain de recherche sur les Grilles de calcul. Globus est devenu aujourd'hui une des infrastructures les plus utilisées dans les projets de Grilles. C'est un produit Open Source qui consiste en un ensemble de composants qui définissent les services de base et les fonctionnalités requises pour construire une Grille informatique. Cela inclut la sécurité, la localisation et la gestion des ressources, la gestion des données, la réservation et les communications. Globus regroupe une série de briques de bases :

- Grid Resource Allocation Manager (GRAM): Utiliser pour l'allocation, la supervision et le control des ressources de calculs.
- GridFTP est une extension du FTP, il contient des fonctionnalités comme la gestion du parallélisme dans le transfert à grande vitesse etc.
- Grid Service Infrastructure : fournit des services d'authentification et de sécurité.
- Global Access to Secondary Storage : accès distant via des interfaces parallèles.

- Globus Executable Management : la construction, la localisation et la mise en cache des exécutables.
- Globus Advanced Reservation and Allocation : pour la réservation et l'allocation des ressources.

Le développement du middleware a débuté à la fin des années 90 et en est maintenant à la version 4(GT4) [FOS 05]. A partir de la version 3(GT3), Globus est basé sur l'architecture OGSA conforme aux recommandations OGSI. Il d'agit d'une encapsulation des services Globus dans des Grilles Services qui rend l'accès à distance plus homogène et répondant à un standard.

3 Environnements Grilles pour le DataMining

Dans les dernières années, plusieurs solutions Grilles ont été développées pour les processus complexes de DataMining. La plus part des approches de DataMining dans les Grilles proposées suivent le modèle de l'Architecture Orienté Service depuis que la communauté Grille est orientée vers le modèle de services [FOS 02] en définissant l'OGSA pour permettre la création et la maintenance des différents services offerts par plusieurs organisations.

L'idée était de déployer un ensemble de services Grilles de DataMining dans une Grille qui peut être composé et ordonnancé d'une manière flexible et efficace pour l'extraction distribuée des connaissances. Les processus d'extraction de connaissances nécessitent la création et la gestion des workflow complexes dynamiques et multi-étapes. Un workflow dans les applications de DataMining est vu comme une série d'opérations de transformations des données qui consistent en l'accès et la préparation de données, la modélisation et la visualisation des modèles (motifs) de DataMining à la fin.

L'Architecture Orienté Service permet l'assemblage d'applications à travers des parties sans se soucier de leurs détailles d'implémentations ou de leurs emplacements. Dans la spécification OGSA, chaque ressource (ordinateur, volume de stockage, programme,..) est représentée sous forme de Service Grille : un Service Web conforme à un ensemble de conventions avec un Support d'interfaces standards.

A travers le WSRF, il est possible de définir des services basiques pour le Support des tâches de DataMining distribuées dans les Grilles. Ces services peuvent adresser tous les aspects impliqués dans le DataMining et les processus de découverte de connaissances à partir de la sélection et le transfère des données jusqu'aux analyses, la représentation et la visualisation des connaissances. Pour cela, il est nécessaire de définir des services correspondants aux étapes individuelles qui composent un processus de découverte de connaissances (prétraitement, filtrage et visualisation) et aux tâches individuelles de DataMining (classification, clustering et les règles d'associations). Cette collection de service de DataMining peuvent constitués une plateforme permettant aux développeurs de concevoir des processus de KDD distribués par la composition des services individuels accessible à travers une Grille. Nous présentant dans ce qui suite les principaux environnements Grilles

développer pour le Support des applications de DataMining et les principaux caractéristiques de ces systèmes.

3.1 The Knowledge Grille :

Knowledge Grille [CAN 03 ; CON 07] est une plateforme Grille conçu pour supporter le DataMining Distribué dans les environnements Grilles. La plateforme a été développée, en utilisant Globus [FOS 05], comme une collection de Services Grille respectant le model de l'Architecturale Orienté Service.

La plateforme utilise les services basiques de la Grille pour construire des services plus spécifiques supportant les tâches de DataMining distribuées et permettent aux utilisateurs d'implémenter des applications de DataMining en utilisant les données, les logiciels et les ressources de calcul situés dans des sites distribuées de la Grille. La plateforme définit les mécanismes et les services de haut niveau qui permettent la publication et la recherche d'informations sur les ressources, la représentation et la gestion des applications DataMining.

A la base, l'architecture de Knowledge Grille est composée de deux groupes de services, classifiés selon leurs rôles et leurs fonctionnalités basiques. Le premier groupe regroupe les services de gestion des sources de données : les outils et les sources de données employés dans le processus entier. Le deuxième groupe est les services de gestions d'exécutions des flux de connaissances, c'est-à-dire les séquences d'étapes qui doivent être exécutés pour effectuer un processus complet de découverte de connaissances en exploitant les avantages des environnements Grilles. Cette approche peut être décrite à travers l'architecture en couche illustrée dans la figure 4.1 qui représente la relation entre les services basiques de la Grille, les services de Knowledge Grille, les services d'analyses de données et les applications KDD.

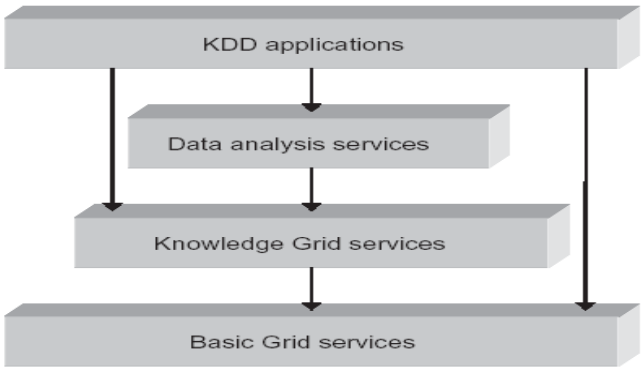


Fig. 3.2 Approche en couche de Knowledge Grid

- **Les Services basiques de la Grille :** ce sont les fonctionnalités fondamentales fournit par un environnement standard de Grille. Ces fonctionnalités inclus les services de sécurité (mécanismes pour l'authentification, l'autorisation, la cryptographie, etc.), les services de management des données (l'accès et le transfère des données, la gestion de répliques, etc.) et les services d'informations (la représentation, la découvert et la supervision de ressources).

- **Les services Knowledge Grille** : ce sont les services conçus spécifiquement pour supporter l'implémentation des services et d'applications de DataMining. Ils inclus les services de gestion des ressources qui fournissent des mécanismes pour décrire, publier et récupérer les informations sur les sources de données, les algorithmes de DataMining et les ressources de calculs, et les services de gestion d'exécution qui permettent aux utilisateurs de concevoir et d'exécuter des applications DataMining distribuées.
- **Les services d'analyse de données** : ce sont les services ad-hoc qui exploitent les services Knowledge Grille pour fournir des fonctionnalités haut niveau d'analyse de données. Un service d'analyse de données peut exposer soit une seule tâche de prétraitement de données ou une tâche de DataMining (classification, clustering, etc.), ou une tâche plus complexe de découverte de connaissance (classification parallèle, méta-Learning, etc.).
- **Les applications KDD** : ce sont les applications de découverte de connaissances construites au-dessus des fonctionnalités fournit par les environnements de Grilles sous-jacent, la plateforme Knowledge Grille ou des services d'analyse de données haut niveau. Des modèles différents, des langages et des outils peuvent être utilisées dans cette Framework pour concevoir des applications KDD distribuées.

Cette architecture flexible permet la conception d'application de découverte de connaissances (KDD) construites au-dessus des services d'analyses de données et au-dessus des services Knowledge Grille. En outre, ces services peuvent être composés ensembles avec les services basiques fournit par un middleware de Grille générique pour développer des applications KDD.

Les travaux précédents sur Knowledge Grid été concentrés sur le développement du système en utilisant les derniers middlewares de Grilles [CAN 04b] et aussi sur la conception et l'évaluation des applications DataMining distribuées [BUE 04 ; CAN 04a]. Le WSRF a été adopté pour la ré-implémentation des services de Knowledge Grille selon le model de l'Architecturale Orienté Service [CON 07]. Ces services seront utilisés soit pour exécuter des applications de DataMining distribuées ou pour construire des services d'analyse de données plus spécifiques, qui peuvent à leurs tours être exploités pour composer et exécuter des applications DataMining de haut niveau.

3.2 GridMiner

La plateforme GridMiner [BRE 05b] a été développée dans le cadre d'un projet de recherche dans l'université de Vienne. Le but du projet était de traiter toutes les tâches du processus de découverte de connaissances dans les Grilles et de les intégrés dans une application de Grille Orienté Service. Le système GridMiner offre un environnement robuste et fiable pour le DataMining haut performant et pour les processus OLAP dans les Grilles. La plateforme GridMiner repose sur deux points principaux :

- **Les outils et les technologies:** ils sont destinés pour bien assister les développeurs d'applications à développer des applications d'analyses de haute performance dans les environnements Grilles. La plateforme GridMiner comporte des services de DataMining séquentiels, parallèles et distribués, des services de TextMining, des services OLAP (on-line analytical processing), des services d'intégration de données et des services de visualisation des résultats. Ces services sont intégrés dans des workflows interactives, qui peuvent être dirigé à partir des bureaux ou des diapositives mobiles.
- **Les cas d'utilisations:** ils permettent de montrer comment les outils et les technologies travaillent ensemble et comment peuvent-ils être utilisés dans des situations réelles.

La coopération interactive des différents services dans l'architecture GridMiner (l'intégration, la sélection et la transformation des données, la fouille de données, l'évaluation des motifs et la représentation des connaissances) est la clé pour des analyses productives. Des algorithmes de DataMining séquentiels, parallèles et distribués sont proposés et implémentés sous forme de services Grilles :

- **Les arbres de décision** : une version distribuée de Service Grille capable d'effectuer une classification de haute performance basée sur l'algorithme SPRINT [SHA 96]. Le service est implémenté comme un service OGSA avec état capable de traiter et de distribuer les données issues d'une source de donnée représentée par l'interface OGSA-DAI. L'algorithme et les dérailles de l'implémentation du service sont présenté par [BRE 05a].
- **Les motifs séquentiels** : implémentation de l'algorithme d'extraction des séquences SPADE.
- **Classification des textes :** une implémentation parallèle de l'algorithme d'arbre de classification opérant sur une collection de texte sous format XML. Les détailles d'implémentation et les résultats de performances peuvent être trouvés dans [JAN 06].
- **Le Clustering** : un service permettant la découverte de clusters par l'utilisation de l'algorithme k-means et l'évaluation du modèle final. Le travaille sur le service est en cours.
- **Les réseaux de neurones** : une version parallèle/distribuée de l'algorithme de propagation arrière (back propagation) est implémentée comme une application de haute performance en langage Titanium (un dialecte parallèle de Java) et complètement testé sur des clusters HPC. L'algorithme utilisé, les détailles d'implémentations et les résultats de performances peuvent être trouvées dans [BRE 06].
- **Les règles d'associations** : un service spécialisé opérant au dessus d'OLAP et utilisant les structure de données en Cubs comme des sources de données fondamentales pour l'extraction des règles d'associations. Une description détaillée de la conception et l'implémentation du service peut être trouvé dans [ELS 05]. Ce service sera détaillé dans la section suivante de ce chapitre.

Chaque service peut être employé soit de manière autonome ou comme un module pour construire des services distribués opérant sur des entrepôts de données intégrés dans la Grille et qui peuvent être contrôlés par un moteur de workflow.

3.3 Les services ADaM

Le système ADaM (Algorithm Development and Mining System) [HIN 00] a été originalement développé dans le but de la fouille des grandes sources de données scientifiques de détection de phénomène géophysiques. La conception originale été un système complet qui comporte des composants logiciels clés pour le calcul distribué incluant un démon pour traiter les requêtes de fouille de données, une base de données de fouille pour chercher et monter les données appropriées, un Ordonnanceurs pour ordonnancer les différents jobs de fouilles de données, un ensemble d'opération de fouille de données et un moteur pour analyser les plans ou les workflow de DataMining.

Avec l'arrivée du paradigme de l'Architecture Orienté Service, le système ADaM a été reconçu comme un middleware complet qui peut être utilisé pour des solutions personnalisées et des applications générales, et de fournir des services de DataMining. Il a été restructuré sous forme de composants autonomes qui peuvent être facilement packagé sous forme de services Grilles [RUS 05] (Les algorithmes de DataMining sont packagés comme des exécutables indépendants).

Les services ADaM permettent aux scientifiques d'utiliser les composants dans la fouille de données et le traitement d'image en mode distribuée. Les scientifiques peuvent mixer et relier plusieurs services ADaM avec d'autres services en utilisant différent composeur de workflow pour les workflow de DataMining et les analyses complexes.

3.4 FAEHIM

FAEHIM [ALI 05] est une plateforme pour le DataMining qui permet la composition des services web, avec l'environnement de workflow largement déployés Triana [TAY 07]. La plupart des services web sont dérivés de la librairie d'algorithmes de DataMining Weka [WIT 05].

Les auteurs définissent une plateforme distribuée pour la découverte de connaissance sous forme d'un système de calcul qui fournit un ensemble complet et convenable d'outils haut niveau pour la découverte de connaissances. La plateforme permet aux utilisateurs de définir leurs problèmes, de choisir leurs solutions stratégiques, d'interagir et de gérer les ressources distribuées, de visualiser et d'analyser les résultats, d'enregistrer et de coordonner les tâches de découverte de connaissances. La plateforme consiste en un ensemble de services DataMining exposés via une API, un ensemble d'outils et un système de workflow sont utilisés pour l'interaction et l'assemblage de ces services pour résoudre des problèmes spécifiques de DataMining. Les auteurs présentent une architecture distribuée où les requêtes de DataMining sont exécutées sur des machines distribuées via des services web distribuées.

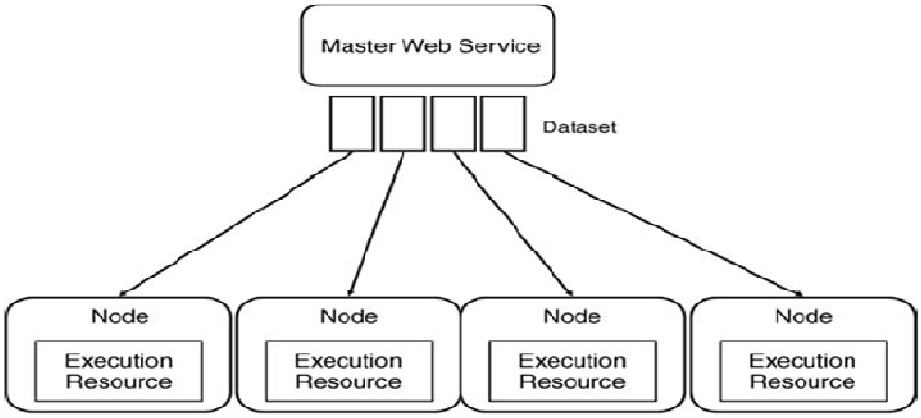


Fig. 3.3 Architecture Web Services distribués

Les éléments clés de cette architecture distribuée sont : **le Service Web Maître** et **les Services Web sur les Nœuds**. Le Service Web Maître accepte les requêtes de DataMining et distribue le travail sur les Services Web des Nœuds en fragmentant les données en entrés et en envoyant chaque partition à un nœud exécutant une instance du Service Web de DataMining sollicité.

Les nœuds renvoient leurs résultats d'exécution au Service Web Maître qui fusionne ces résultats et les renvois à l'utilisateur. Le Service Web Maître implémente les fonctionnalités nécessaires pour l'équilibre de charge et le contrôle d'erreurs.

Les auteurs utilisent les algorithmes de DataMining issus de l'outil GridWeka [KHO 04] qui est une modification de Weka qui lui permettant d'utiliser les ressources de plusieurs ordinateurs.

3.5 Discovery Net

La plateforme Discovery Net [SAI 03] a été construite autour d'un modèle de workflow pour l'intégration des sources de données distribuées et des outils d'analyses. La plateforme est conçue pour le Support de l'exécution des tâches de DataMining distribuées dans un environnement Grille. Le système a été originalement développé comme une partie du projet e-science du Royaume-Uni (2001–2005) [ROW 03] dans l'objectif de produire une plateforme de haut niveau orienté-application permettant aux scientifiques (utilisateurs-finals) la dérivation de nouvelles connaissances à partir des dispositifs, des capteurs, des bases de données, des composants d'analyse et des ressources de calcul à travers le réseau Internet et les Grilles.

Cependant, et à travers les années, le système a été utilisé dans de nombreux projets scientifiques académiques et industriels de DataMining. Cela inclus des applications de sciences naturelles [GHA 02, 05 ; LUQ 06], de surveillance environnemental [RIC 06] et d'applications de modélisation géo-risque [GUO 05].

Plusieurs idées de recherche développées dans le système sont aussi incorporées dans le système KDE InforSense : un système commercial de gestion des workflow et de DataMining qui est largement utilisé dans les applications orienté Business.

3.6 DataMiningGrid

DataMiningGrid [STA 08] est un environnent extensible, basé sur les standards pour le Support d'applications de DataMining dans les Grilles. Le projet DataMiningGrid est un investissement de l'Union Européens ayant pour objectif le développement d'un système générique facilitant le développement et le déploiement d'applications de DataMining dans les Grilles. Le résultat principal du projet été une collection de composants logiciels constituant le système DataMiningGrid. Le système DataMiningGrid forme un système à jour pour le DataMining distribué basé sur une Architecture Orienté Service et conçu pour répondre aux exigences des scénarios modernes de DataMining distribuées. Le système est basé sur les technologies open-source et les standards communs comme Globus, WSRF et OGSA. Le système est basé sur une architecture en couche comprenant les couches suivantes : les ressource matériels et logiciels, le middleware de la Grille, les services DataMiningGrid haut niveau et la couche client.

Le système offre une large applicabilité grâce à son mécanisme extensible qui permet d'ajouter à l'environnement Grille des composants tiers de DataMining sans modification programmatique des composants ou de l'environnement Grille [WEG 07]. Le système DataMiningGrid a été étendu par un ensemble de composants pour le Support du partitionnement des données en entrés ou pour le traitement des résultats dans un environnement de type Grille. L'interface utilisateur principale du système est l'éditeur et le manager de workflow Triana [TAY 07], qui a été étendu au cour du projet par des composants spécifiques facilitant l'accès aux environnements Grilles.

Dans le contexte du système DataMiningGrid, une application est définit comme un fichier exécutable automne (C, Python, Java) qui peut être lancé en ligne de commande. Les dérailles de l'architecture du système et le processus d'adaptions aux Grilles sont décrit dans [STA 08].

3.7 DMGA

DMGA (Data Mining Grille Architecture) [SAN 04; PER 07] est une architecture générique pour le déploiement des services DataMining dans les Grilles. Cette architecture est conforme à la vision actuelle des Grilles basée sur OGSA, et définissant des services reflétant les principales étapes du processus de DataMining.

L'architecture DMGA étende la spécification OGSA par la définition de nouveaux services DataMining utilisant les services basiques de la Grille. Elle permet la composition des services par la création des workflows qui permet à plusieurs services d'être ordonnancés de manière flexible et efficace. La composition de ces services peut être effectuée de deux manières :

- **La Composition horizontale :** Une tâche de DataMining complète peut impliquer la collaboration de plusieurs services fonctionnels. Ces services de fonctions différentes (prétraitement, la fouille des données et le poste-traitement) sont combinés et exécuter pour compléter une telle tâche. Typiquement, la sortie d'un service correspond à l'entrée du service suivant.
- **La composition verticale:** Chaque service étant assigné à une partition de données spécifique, ces services ont la même fonctionnalité mais accèdent à différentes portions de donnée qui peuvent être géographiquement distribuées. Les résultats issus de toutes les ressources sont combinés pour calculer la réponse, qui est par la suite transférée au client.

Une implémentation de l'architecture DMGA, appelé WekaG [PER 05], a été développée en basant sur Weka [WIT 02].

3.8 Discussion

GridMiner a été conçu pour le Support du DataMining et des traitements OLAP dans les environnements de calculs distribués. GridMiner implémente un nombre d'algorithmes de DataMining, certains en versions parallèles, en plus de quelques tâches de TextMining. L'architecture orienté-service du système facilitant l'intégration et l'exécution des services comme des workflows, chaque service DataMining est implémenté comme un service Grille autonome spécifié par OGSA.

Knowledge Grille (K-Grille) est une plateforme de haut niveau, orienté-service fournissant des outils et des services de DataMining à base de Grille. Le système K-Grille facilite un grand rang de tâches de DataMining en plus des tâches de gestion des données et de représentation des connaissances. L'architecture du système est organisée en services hauts niveaux et en services du niveau noyau construit au dessus de la couche des services basiques de la Grille. Les composants principaux de K-Grille sont implémentés en utilisant l'environnement VEGA (Visual Environment for Grille Applications), qui lui-même basé sur Globus. Des développements récents cherchent à ré-implémenter la version antérieure du système en une version accommodé au WSRF. K-Grille n'est pas disponible en open-source.

DMGA est une architecture Grille flexible composé de services Grilles générique de DataMining. WekaG est une implémentation de cette architecture basée sur Weka et Globus. Le principale avantage de la combinaison DMGA/WekaG est que DMGA/WekaG peut être adapté aux besoins des processus complexes de DataMining et que le système est facile à utilisé par les utilisateurs de DataMining (les interfaces utilisateurs de Weka et WekaG sont les mêmes). De plus, les nouveaux services DataMining peuvent être ajoutés de façon flexible et il est possible d'utiliser une combinaison de plusieurs services DataMining. Finalement, WekaG supporte les implémentations parallèles des algorithmes de DataMining. L'implémentation de WekaG (qui est basé sur Weka) est restreinte aux applications implémentées dans Weka. Weka est disponible en Open Source et WekaG probablement va être en open source dans le futur.

FAEHIM implémente une plateforme basé sur l'approche des services Grilles pour le Support de DataMining. La plateforme consiste en des services Grilles de DataMining est un moteur de workflow permettant aux utilisateurs de composer ces services pour la résolution d'un problème particuliers. Trois types de services Grilles sont fournit pour implémenter les fonctions de DataMining : la classification, le clustering et les règles d'associations. Les algorithmes utilisés pour implémenter les fonctions de DataMining sont pris de Weka. De plus, des services Grilles de visualisation basée sur GnuPlot et Mathematica sont aussi fournit pour la visualisation de la sortie des services Grilles de DataMining. Les sources de données peuvent être lues par un service Grille à partir d'un espace de fichiers local ou à partir d'un site distant. FAEHIM utilise Triana pour définir les processus de DataMining.

Discovery Net fournit un model de calcul orienté-service pour la découverte de connaissances. Il permet aux utilisateurs d'accéder et d'utiliser les logiciels d'analyse de données et les sources de données des parties tiers. Le système est basé sur Globus et fournit les composants : (a) pour déclarer les propriétés des composants logiciels d'analyses et les sources de données scientifiques, (b) pour retirer et composer les servies de découvertes de connaissances, (c) pour intégrer les données structurés et semi-structurés à partir des différents sources de données utilisant des schémas XML, et (d) pour déployer et publier les procédures de découvertes de connaissances comme de nouveaux services. Le système fournit aussi le langage DPML (Discovery Process Markup Language) qui est une représentation à base de XML des workflows de Discovery Net. Le système permet aussi l'accès dynamique et l'intégration de plusieurs sources de données dans le workflow. Pour l'intégration des données, des interfaces pour les bases de données SQL et des sources OGSA-DAI sont construites. Discovery Net ne semble pas supporter WSRF.

Le system ADaM contient une grande variété de composants de DataMining et de traitement d'images conçus pour les analyses des données scientifiques et des données des capteurs distants. Les composants du système peuvent être configurés pour créer des processus de DataMining personnalisés. Dans ces derniers versions, les opérations de DataMining individuelles de ADaM (appelé librairie) sont disponibles sous forme des exécutables, des librairies C/C++ ou des modules (Python), et ils peuvent être accédés via des interfaces externes multiples facilitant l'implémentation des composants de DataMining et de traitement d'image comme des services Web ou services Grilles. ADaM concentre sur les tâches de traitement d'images et le système supporte un large rang d'interfaces et de mécanismes de composition permettant de réaliser des processus de DataMining personnalisés. Le système semble manquer des services d'édition et de gestion des workflows appropriés.

Le système DataMiningGrid est basé sur Globus et d'autres technologies et standards comme OGSA-DAI, Triana et GridBus fournissant des outils et des services facilitant l'adaptation d'applications de DataMining aux Grilles sans aucune intervention au niveau de l'application. Le système DataMiningGrid est capable d'accommoder les applications DataMining issus d'un grand rang de plateformes et, de technologies. L'adaptation des programmes de DataMining existants est achevée à travers l'utilisation de métadonnée et de leurs

informations associées et des services de Ressources Brokers. Le système DataMiningGrid est conçu autour des principes de l'architecture orienté-service et supporte le WSRF. Il fournit des interfaces utilisateurs flexibles basées sur la technologie sophistiquée de workflow (Triana) en plus d'une interface de type client web.

4 Déploiement des règles d'associations sur les Grilles

Dans cette section nous présentons quelques travaux d'intégration et d'implémentation de la technique d'extraction des règles d'associations dans quelques environnements de Grilles présentés dans la section précédente et quelques tentatives d'implémentation de cette techniques par certains auteurs sur des environnements de Grille de testes :

- Sanchez et Al ont déployé dans [SAN 08] l'algorithme Apriori sur un environnement Grille DMGA/WekaG afin d'évaluer la composition horizontale proposée dans DMGA. L'algorithme Apriori est exécuté à travers la composition horizontale de deux services : le service AprioriG et le service de transfère de base GridFTP [ALL 01]. Le premier service, inclus dans WekaG, est utilisé pour construire les règles d'associations et le deuxième pour la sérialisation d'objets a transféré entres les nœuds de l'environnement Grille. Une fois le client a contacté le Broker et que ce dernier a choisi les services appropriés installés sur quelques serveurs de la Grille, la fonctionnalité de l'application est subdivisée entre le coté client et le coté serveur. La partie client inclus l'interface utilisateur et la partie serveur est responsable sur l'exécution de l'algorithme Apriori. Dans cette implémention, les auteurs ont utilisé les objets socialisables pour le stockage et la récupération de l'état des objets. Le module client est responsable sur deux tâches :

- ✓ l'interrogation du service GridFTP pour le transfert de données.
- ✓ L'invocation de la fonction de création de règles d'association du service AprioriG.

L'algorithme est exécuté par le service AprioriG et les résultats sont renvoyés à l'interface graphique de l'utilisateur. Les auteurs ont évalué les performances du service AprioriG par rapport à une version d'Apriori implémenté par Weka. Le résultat montre que le temps processeur nécessaire pour le processus de DataMining dans le cas d'une station standard est double au temps nécessaire pour un serveur qui fournit un service Grille.

- Craus et Al présentent dans [CRA 05] l'implémentation de la technique d'extraction des règles d'association en utilisant l'algorithme Apriori au-dessous du Globus. Les auteurs ont proposé une méthode exposée sous forme d'un service Grille et conformément à l'OGSA pour l'intégration de la tâche d'extraction des règles d'associations à partir des bases de données géographiquement distribuées. Le service d'extraction des règles d'associations proposé est interactif avec le reste des services de la Grille : service annuaire, service de création, d'autorisation, de notification, de gestion et de concurrence.

Les auteurs présentent un cas d'étude pour le déploiement du service Grille sur une possible infrastructure Grille d'une organisation virtuelle : L'entreprise est composée d'une succursale

centrale et des succursales locales (LB). Chaque succursale est constituée d'un nombre de nœuds de la Grille (GN) interconnectés dans une infrastructure de Grille (figure 3.4). L'implémentation de l'infrastructure Grille est basée sur Globus et on veut lancer la tâche d'extraction des règles d'associations à partir de la succursale centrale pour la fouille des données de la succursale centrale et des succursales locales.

L'environnement sous-jacent de la succursale centrale encapsule des ressources de calculs et de stockages pour la création des services d'extraction de règles d'associations et le stockage des connaissances collectées. L'environnement implémente aussi un service d'annuaire d'une organisation virtuel pour fournir des informations sur la localisation de tous les services de fouilles, de transformations des données et de la base de données. Chaque fournisseur de service à un service d'annuaire local (LR) qui fournit des informations sur l'interface du service implémenté.

L'utilisateur de l'application interroge l'annuaire de l'organisation virtuel pour rechercher les services de d'extraction des règles associations. Cette requête crée des instances de services APRIORI sur les différents environnements hôtes. Les nouvelles instances de services crées commence l'interrogation des données à partir des services de base de données et place les résultats intermédiaires dans l'espace de stockage local. Les services de DataMining utilisent les mécanismes de notifications pour fournir à l'utilisateur des mises à jour périodiques sur leurs statuts. L'utilisateur reçoit une notification lorsque le service d'extraction des règles d'associations termine ces jobs et que les résultats peuvent être explorés à partir de la base de connaissances.

Le service Grille d'Apriori implémente une version séquentielle d'Apriori. À la première étape, l'algorithme cherche tous les 1-Itemsets candidates et tous les 1-Itemsets fréquents. Par la suite l'algorithme cherche tous les 2-Itemsets fréquents, ensuite l'algorithme génère un arbre de hachage pour le stockage des k-Itemsets. L'algorithme multiple les parcours des bases de données et ne les charges pas en mémoire ce qui est très utile pour des bases de données volumineuses.

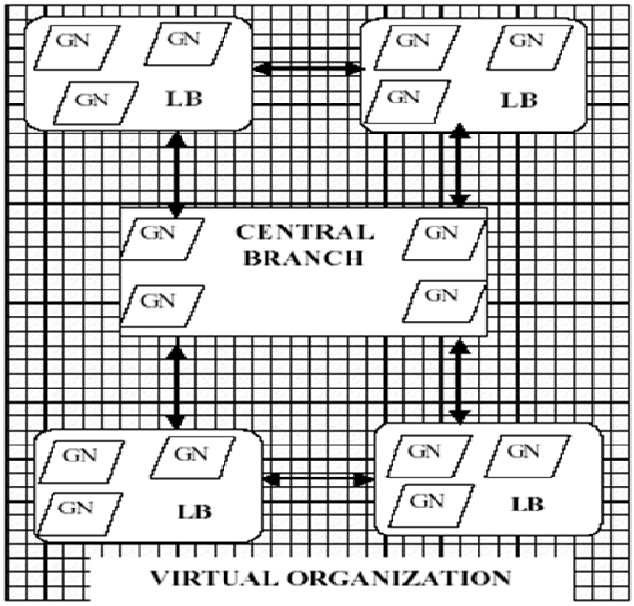


Fig. 3.4 Infrastructure d'une organisation virtuelle utilisant la technologie des grilles

Les expérimentations sont effectuées sur un réseau d'Intranet qui connecte trois LANs. Les données transactionnelles sont stockées sur trois SGBD différents (Oracle 10g, PostGreSQL et MySQL). Une série de testes sont exécutés pour comparer l'implémentation de Apriori en Service Grille avec une implémentation classique (librairie Weka) de l'algorithme Apriori. Les résultats montrent de meilleures performances pour la version Grille.

- Luo et Al présentent dans [LUO 06] un nouvel algorithme pour l'extraction des Itemsets fréquents maximaux à partir des bases de données sur une Grille de données. Cet algorithme, appelé GridDMM (Grid-Based Distributed Max-Miner) est une version parallèle de l'algorithme séquentiel d'extraction des Itemsets maximaux Max-Miner et qui est adapté pour l'utilisation sur les environnements de Grilles. L'algorithme a été implémenté dans un système de Grille de données à base d'un cluster en utilisant Globus et MPICH-G2. La base de données est partitionnée en *n* partitions $\{D_0, D_1, D_2, \ldots, D_{n-1}\}$, chacune réside dans un nœud de calcul $\{P_0, P_1, P_2, \ldots, P_{n-1}\}$ de manière que chaque nœud à le même nombre de transactions. L'algorithme se déroule en deux phases : la phase d'extraction locale et la phase d'extraction globale.

  - **La phase d'extraction local:** Chaque nœud de calcul Pi (0<i<n-1), exécute l'algorithme Max-Miner séquentiel sur sa partition de données locale Di pour calculer les Itemsets fréquents maximaux locaux (LMi). Par la suite tous les LM0, LM1,…, LMn-1 sont échangés entre les nœuds pour générer l'ensemble initial des candidats globaux GC1 pour la première passe de la phase globale.
  - **La phase d'extraction Globale:** Dans cette phase, l'algorithme effectue une recherche de Haut en Bas sur l'ensemble des candidats en utilisant un arbre de préfix. Chaque nœud commence la collecte des supports locaux des candidats globaux en

parcourant sa partition Di et en échangeant ces supports avec les autres nœuds pour le calcul des Itemsets fréquents maximaux globaux. Les Itemsets candidats globaux fréquents sont inclus dans un ensemble nommé FrequenSet et ceux qui ne le sont pas fréquents sont insérés dans un ensemble nommé InfrequentSet. En suite tous les (k-1)-sous-ensemble des Itemsets de l'ensemble InfrequentSet sont considérés comme de nouveaux candidats globaux pour l'itération suivante. La phase se poursuit jusqu'à ce que l'ensemble global des candidats soit vide.

L'algorithme utilise un schéma de communication logique en *n-Cubs* pour effectuer les communications entre les nœuds. Dans une structure *n-Cub*, il y a 2^n nœuds et chaque nœud a une adresse binaire de *n* bits et il a *n* nœuds voisins directement liés. Ainsi les 2^n nœuds peuvent échanger les supports locaux en *n* étapes. De plus, les auteurs ont développé un nouvel arbre de préfixe qui facilite le stockage et le calcul des supports des candidats globaux de longueurs différentes. Les auteurs utilisent trois techniques pour la réduction de la taille de l'ensemble des candidats globaux à savoir : l'estimation des supports globaux, l'élagage basé sur les sous-ensembles non fréquents et l'élagage basé sur les sur-ensembles non fréquents.

Pour la première technique, lors de la fusion des items fréquents maximaux locaux (IFMs) de tous les nœuds de calcul, les supports des IFMs locaux qui ont leurs sur-ensembles fréquents dans d'autres nœuds permettront d'estimer le Support minimal des candidats dans ces nœuds : Le Support minimal estimé d'un candidat est le plus grand Support de tous ses sur-ensembles. Pour les deux autres techniques, l'algorithme utilise les ensemble des fréquents (FrequenSet) et les non fréquents (InfrequentSet) pour l'élagage. Après chaque étape dans la phase d'extraction globale on détermine pour chaque candidat global s'il est fréquent maximal ou non. Si un candidat est fréquent, on le mettra dans l'ensemble FrequenSet seulement si aucun de ses sur-ensembles n'est déjà dans cet ensemble. D'autre part, si le candidat global est non fréquent, on le mettra dans l'ensemble InfrequentSet seulement si aucun de ses sous-ensembles n'est déjà dans cet ensemble. Ces deux techniques d'élagages permettent de réduire l'ensemble des candidats globaux de manière considérable pour chaque étape. Maintenir ces deux ensembles sans aucune redondance est important pour obtenir un élagage efficace.

Les expérimentations sont effectuées sur un système de Grille de données composé de neuf (09) nœuds de calcul en utilisant Globus et (MPICH-G2). Durant la phase de fouille, le nœud de calculs accède aux partitions de base de données à travers des appels Entrées/Sorties MPICH-G2, les fonctions d'envoi/réception de MPICH-G2 sont utilisées pour le transfert des données intermédiaires générées et suivant le schéma logique en *n-Cub*.

Les résultats montrent que cet algorithme est performant sur ce type de système. L'algorithme réduit considérablement le coût de communication grâce au schéma *n-Cub* utilisé. L'algorithme génère également un ensemble de candidats réduits en se basant sur les trois techniques d'élagages précédents. Cet algorithme est plus scalable en termes de taille de base de données et en termes de nombre de nœuds.

- Sakthi et Al présentent dans [SAK 08] l'implémentation du processus d'extraction des règles d'associations sur KNOWLEDGE GRILLE sous forme d'un service Grille. La Grille est utilisée comme une plateforme pour le déploiement des services des applications d'extraction de connaissances et de gestion des connaissances. Un processus de préparation de données est effectué initialement pour la collecte et le nettoyage des sources de données. Le service Apriori est implémenté au-dessous de la couche du niveau supérieure de la Grille K-Grille.

Le service Grille Apriori est implémenté sur chaque nœud de la Grille. L'algorithme Apriori local parcours la base de données locale Di pour générer l'ensemble des 1-Itemsets fréquents dans la base locale. Les 1-Itemsets fréquents locaux sont envoyés aux autres nœuds de la Grille par l'utilisation de MPICH-G2 pour trouver les 1-Itemsets fréquents globaux. Chaque nœud de la Grille reçoit les 1-Itemsets fréquents de tous les autres nœuds de la Grille et finalement il mis à jour sa base locale par les 1-Itemsets fréquents globaux. L'algorithme se termine avec l'ensemble des Itemsets fréquents enregistrés dans tous les nœuds de la Grille. Ces Itemsets sont enregistrés dans une base de connaissances (Knowledge Base Repository) et peuvent être utilisés pour générer les règles d'associations.

Différentes configurations sont utilisées pour évaluer l'algorithme Apriori parallèle. Des tests sont effectués sur un PC autonome et deux clusters de trois nœuds chacun et trois clusters de trois nœuds chacun. Sur chaque nœud est installé Globus (GT3) et le service Apriori est déployé sur ce nœud. Les bases de données de fouille sont stockées sur trois SGBD différents : Oracle 10g, PostGreSQL and MySQL.

Les tests effectués montrent que le service Grille Apriori apporte plusieurs bénéfices en temps d'exécution et qu'une accélération linéaire est enregistré par rapport au nombre de nœuds de la Grille en plus d'une bonne scalabilité en terme de taille de données.

- Elsayed et *Al* traitent un problème un peu spécial dans [ELS 05] et implémentent la technique d'extraction de règles d'associations dans la plateforme GridMiner. Le but des auteurs était de développer un moteur d'extraction de règles d'associations à base d'OLAP qui peut être utilisé comme un Service Grille dans GridMiner. L'utilisation des structures de données en cubes OLAP en préface a une tâche de DataMining est référé en littérature comme la fouille analytique enligne (OLAM : On-Line Analytical Mining). La construction des cubes de données à partir des grandes sources de données est considérée comme une importante étape de prétraitement de DataMining où les fonctions de DataMining peuvent être intégrés avec les opérations OLAP pour des extractions de connaissances interactives et à plusieurs niveaux d'abstractions.

La plateforme OLAM est constituée d'un moteur OLAP et d'un moteur d'extraction des règles d'associations (ARM Engine). Les auteurs ont fournit deux variantes de plateformes OLAM haute performance à base de Grille (un simple prototype séquentiel et une solution parallèle et distribuée). Dans le prototype séquentiel, le moteur d'extraction des Règles d'Associations est incorporé dans le moteur OLAP comme un outil additionnel de DataMining qui peut être appliqué sur les cubes construits par le moteur OLAP. Dans la

version parallèle/distribuée, le moteur d'extraction des règles d'associations est fournit séparément du moteur OLAP.

Le moteur d'extraction des règles d'associations a pour sortie un document PMML[1], un standard de la communauté d'extraction de connaissances et de DataMining, qui peut être passé à un Moteur de visualisation ou à une interface utilisateur graphique.

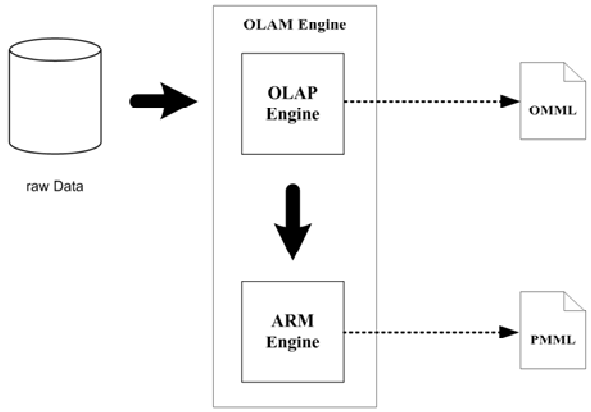


Fig. 3.5 La Framework générale du moteur OLAM

L'extraction des règles d'associations à base d'OLAP est composée des principales étapes : la génération du cube de données multidimensionnelles, l'extraction des Itemsets fréquents et la génération des règles d'associations. Les auteurs adoptent une approche similaire à la méthode itérative utilisée dans Apriori pour le calcul des Itemsets fréquents, les 1-Itemsets fréquents sont calculés en premier, et ils sont utilisés en suite pour trouver les 2-Itemsets fréquents et ainsi de suite. Cette méthode est implémentée dans la plateforme OLAM séquentielle et parallèle/distribuée comme suite :

- **La version séquentielle:** Initialement, on récupère tous les 1-Itemsets candidats qui sont les valeurs distinctes de chaque dimension. On construit pour chaque dimension du cube une Hach-Map contenant les items de la dimension correspondante. On interroge les cellules du cube pour chaque 1-Itemset pour calculer son Support. Si le Support de l'Itemset n'est pas inférieur du seuil *Minsup* alors l'Itemset est fréquent. A la fin de la première itération de l'algorithme, l'étape d'élagage est effectuée en supprimant tous les 1-Itemsets qui ne sont pas fréquents.

  La deuxième passe de l'algorithme calcule les 2-Itemsets fréquents. On génère d'abord toutes les combinaisons possibles des 2-Itemsets candidats et on calcule ensuite leurs supports. Une fois les 2-Itemsets fréquents trouvés, l'étape d'élagage est encore lancée en supprimant les 2-Itemsets non fréquents. L'algorithme continue jusqu'à ce qu'on trouve les n-Itemsets fréquents.

- **La version distribuée** : Dans la Framework OLAM parallèle, le moteur d'extraction des règles d'associations est implémenté séparément du moteur OLAP et communique avec ce dernier à travers des requêtes. L'idée de l'OLAM parallèle et distribué est de construire des cubes de données qui peuvent être géographiquement distribuées à

travers un ensemble de nœuds d'une Grille et qui apparaît à l'utilisateur final ou à une application donnée comme étant un seul grand cube virtuel. Ce cube virtuel est divisé en plusieurs petites parties appelées Segments de Cubes qui peuvent être divisés davantage en de petits segments qui sont attribués à des nœuds de Grilles. Après le démarrage du moteur OLAP parallèle, un fichier XML(OMML) contenant tous les items est généré. Ce fichier représente la donnée en entrée du moteur d'extraction parallèle des règles d'associations.

A cette instant, on commence la génération d'Itemsets fréquents et à la différence de la version séquentiel où on génère les Itemsets fréquents un par un, les auteurs ont parallélisé la génération des i-Itemsets fréquents (i reflète l'$i^{ème}$ dimension du cube de donnés) en regroupant les Itemsets fréquents candidats par leurs niveau (taille) et par leurs dimension dans des interrogations. Ces interrogations sont traitées en parallèle tout en gardant l'aspect itératif de la version séquentiel où les 1-Itemsets sont utilisés pour générer les 2-Itemsets et ainsi de suite jusqu'à ce que le niveau est égale au nombre total de dimension du cube de données.

Les auteurs ont intégré le moteur OLAM dans GridMiner. Le moteur OLAM est fournit comme un moteur autonome en Java qui sera intégré par la suite dans la plateforme GridMiner. La figure 4.6 illustre un scénario typique d'intégration dans la Framework GridMiner :

- L'utilisateur spécifie les sources de données de formats différents (fichier XML, fichier CST, base de données MySQL et ORACLE) situés à des endroits géographiques différents et définit l'intégration de ces sources de données comme la première étape de ce workflow.
- Un Nettoyage des entrées obsolètes dans la source de données composée est effectué.
- Une tâche de DataMining est assignée au nœud OLAP qui reçoit la source de données intégrée et nettoyée en entrée et construit un cube OLAP multidimensionnelle. Le moteur OLAP est capable d'effectuer des opérations OLAP spécifiques (roll-up, drill-down, agrégation).
- L'Extraction des Itemsets fréquents et des règles d'associations dans les cubes OLAP. La sortie ici est un fichier PMML qui peut être utilisé par un outil de visualisation compatible-PMML ou par le service de visualisations de GridMiner.

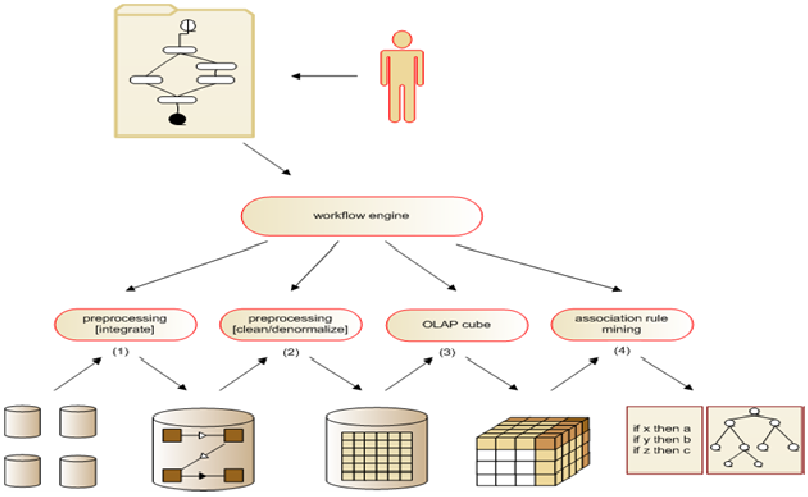


Fig. 3.6 Un workflow OLAP typique de fouilles

Les quartes instances des services composent ensemble un simple processus KDD. Ces instances de services sont initiés sur des nœuds de la Grille appropriés selon la configuration et l'exécution du workflow spécifié dans l'interface utilisateur graphique de GridMiner où l'utilisateur peut créer, enregistrer et charger des workflows.

- Fiolet présente dans [FIO 06] un algorithme distribué pour le problème des règles d'associations sur les Grilles en suggérant l'utilisation d'un partitionnement (horizontal) intelligent des données de manière à obtenir des fragments très indépendants (avoir une similarité maximale des données au sein d'un fragment ainsi qu'une dissimilarité maximale des données entre les fragments). Cette fragmentation intelligente de la base de données permet de réduire la taille de l'espace de recherche visité pour la résolution du problème des règles d'associations et d'améliorer l'efficacité de l'exécution distribuée : Les résultats locaux sont riches sur chaque fragment (nombreuses informations disponibles, qui sont spécifiques aux données du fragment) ; Les résultats globaux peuvent être obtenus au moindre coût à partir des résultats locaux.

Ce partitionnement peut être obtenu par clustering, l'auteur présente donc un nouvel algorithme CDP (Clustering Distribué Progressif) qui exécute un clustering distribué de manière progressive et efficace respectant les contraintes d'exécutions sur un environnement de type Grille. L'auteur introduit un algorithme distribué, appelé DIC-Coop pour le problème d'extraction d'Itemsets fréquents basé sur l'utilisation du partitionnement ‘’intelligent ‘’ issu du clustering. Cet algorithme, inspiré de l'algorithme DIC [BRI 97] permet de faire collaborer des traitements parallèles des fragments, tout en respectant les critères d'exécutions (pas de mémoire partagée, pas de synchronisation). De plus et afin de permettre une distribution optimale, une méta-version DIC-Coop est proposée dans le cas où la fragmentation fournirait des fragments de transactions trop grands pour être traités localement sur un seul site. L'auteur propose de distribuer leurs traitements non plus sur un site (ou nœud de la Grille), mais sur une sous grappe, en utilisant l'algorithme DIC-Coop.

L'algorithme DIC est basé sur une méthode similaire à Apriori et propose une diminution du balayage nécessaire du jeu de données. L'algorithme utilise un principe de fenêtrage de la base : le jeu de données est décomposé en sous-ensemble de taille donnée, et durant chaque itération, un sous-ensemble de transactions est lu. Ainsi, plusieurs ensembles d'Itemsets candidats de tailles différentes sont traités simultanément durant chaque itération, ce qui permet de diminuer le nombre total de balayages du contexte. Des marques sont attribués aux Itemsets permettant de connaitre la catégorie à laquelle ils appartiennent : fréquent confirmé (SB), fréquent potentiel (DB), non fréquent confirmé (SC), non fréquent potentiel (DC). Le parcours du premier bloc permet d'estimer les fréquences des 1-Itemsets. Lorsqu'on a parcouru tout ce premier bloc, on génère les 2-Itemsets candidats à partir des estimations de fréquences réalisées pour les 1-Itemsets. Le parcours du deuxième bloc permet d'améliorer les estimations des fréquences des 1-Itemsets et de débuter l'estimation des fréquences des 2-Itemsets.

L'adaptation de l'algorithme DICCoop repose sur une collaboration entre un site fédérateur F et des sites de traitements T_i exécutant une version modifiée de l'algorithme DIC. Le rôle du fédérateur est de collecter les informations locales sur les sites T_i, de les agglomérer pour en déduire l'information globale et de transmettre les informations globales ainsi construites aux sites de traitement T_i. Le principe de DICCoop est d'utiliser un algorithme inspiré de l'algorithme DIC sur chaque fragment (traitements locaux sur les sites Ti), et d'effectuer des communications de collaboration entre ces sites Ti et le site fédérateur F qui enrichit les traitements des sites locaux.

Différentes modes de communication entre le fédérateur F et les sites Ti ont été comparées : Aucune communication (effectuer des traitements indépendants par l'algorithme DIC classique), tous communiquer (on effectue de communication aussi souvent que nécessaire) et communication paramétrés (Les communications sont déclenchées dans les cas où la quantité d'information à envoyer soit suffisante et qu'un certains délais soit dépassés depuis le dernier envoi). Les expérimentations montrent que la version PCom apparaît comme la solution la plus adaptée à une exécution sur plateforme de type Grille de calcul.

5 Approche proposée pour les Grilles

Actuellement, peu de Framework de recherche existe pour le déploiement des applications de DataMining distribuées sur les Grilles. Certains sont des environnements généraux supportant les tâches de DataMining sur des machines qui appartiennent à une Grille, d'autres sont des tâches individuelles de DataMining spécifique à une application qui sont adaptés aux Grilles, et certains autres implémentations singulières d'algorithmes de DataMining.

La recherche des Itemsets fréquents dans un environnement Grille implique souvent un nombre important de nœuds qui coopère pour la résolution du problème d'extraction des règles d'associations. Nous avons introduit une version distribuée de l'algorithme AprioriDCM qui est adapté à l'utilisation dans un environnement de Grille appelé AprioriGCM (Grid-Based AprioriCM).

5.1 Présentation de L'algorithme AprioriGCM (Grid-Based AprioriCM).

La Grille est une infrastructure de calcul distribué qui facilite la coordination et le partage des ressources dans de différentes organisations et des domaines administratives à travers des sites géographiquement dispersés. Un domaine administratif est une collection de hôtes et de routeurs, et les interconnexions réseaux, gérer par une seule entité administrative. La figure suivante reflète une architecture qu'on trouve dans les Grilles.

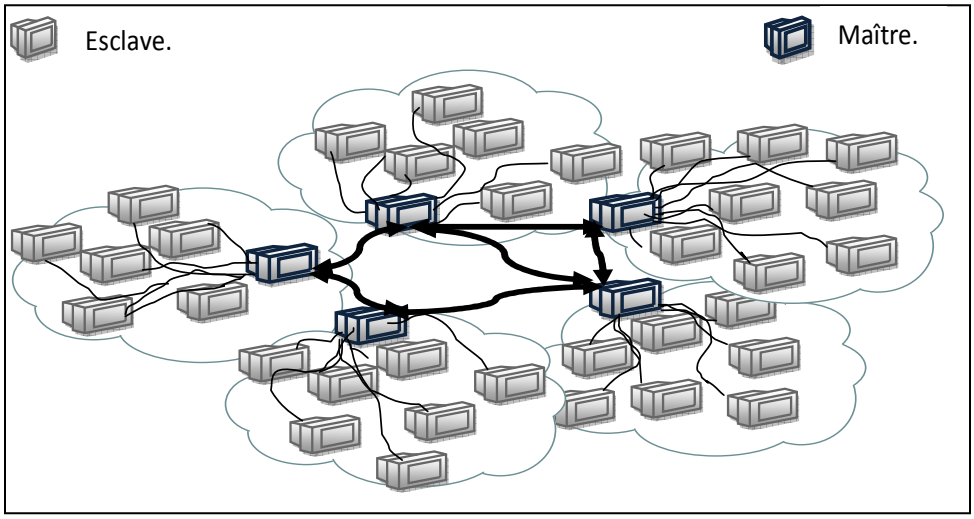


Fig. 3.7 Architecture proposé pour la grille

La version distribuée AprioriDCM2 (validation à posteriori) présenté auparavant dans le chapitre 2 favorise les calculs indépendants et minimise le nombre de messages et de synchronisations globales nécessaires pour le calcul de l'information globale. Cette version semble la plus appropriée à un environnement de type Grille par rapport à la version AprioriDCM (validation par itération).

Cependant, l'utilisation d'un seul site Maître dans AprioriDCM2 pour la coordination de l'ensemble des nœuds de la Grille impliqués pour l'extraction des Itemsets fréquents pose un problème important : Le site Maître devient un goulot d'étranglement à grand échelle.

Nous avons donc adapté l'algorithme AprioriDCM2 à l'utilisation dans un environnement Grille en segmentant l'ensemble des nœuds en groupes/domaines et en multipliant les sites coordinateurs Maîtres.

L'ensemble des sites participant à la Grille est organisé en groupes/domaines. Chaque ensemble de *G* sites forme un seul groupe/domaine. Dans chaque groupe, on considère deux types de sites : un site Maître et des sites Esclaves.

Le site Maître est désigné dans chaque groupe *G* pour s'occuper du calcul des supports globaux de groupe *G*, le site Maître diffuse ensuite ces informations aux autres sites Maîtres des autres groupes. Le site Maître joue le rôle du site fédérateur du groupe *G* en question. Les sites Esclaves s'occupent des traitements proprement dits.

On suppose que la base de données est géographiquement distribuée entre les sites Esclaves de la Grille. Le schéma général de l'algorithme AprioriGCM est décrit par les étapes suivantes :

- **Etape 1 : Construction de la structure CountMap globale**
  - **Etape 1.1 : construction de CountMap globale du groupe.**

    Chaque site Maître lance l'opération de la construction de la structure CountMap globale dans son groupe :
    - Chacun des sites Esclaves du groupe construit la structure CountMap locale (LCounMap) en parcourant les portions de base de données locales.
    - Le site Maître du groupe construit la structure CountMap globale du groupe dont il est responsable (GCountMap) par la fusion des différents LCountMap des sites Esclaves appartenant à son groupe.
  - **Etape 1.2 : construction de CountMap globale entre tous les groupes.**

    Les sites Maîtres des différents groupes diffusent (broadcaste) entre eux les valeurs des GCountMap globaux pour avoir les valeurs globales de tous les sites participants de la Grille.

A la fin de l'étape 1, chaque site Maître maintient la structure GCountMap globale qui contient les fréquences globales de la base de données entière. Ainsi, chaque site Maître de chaque groupe construit la liste des 1-Itemsets et des 2-Itemsets fréquents globaux en utilisant la structure GCountMap.

- **Etape 2 : Extraction des k-Itemsets fréquents globaux**

Chaque site maître de groupe construit itérativement la liste des k-Itemsets candidats globaux (k≥3) en utilisant le Support$^+$ et la structure GCountMap :

$$\text{Support}^+(Y) = \begin{cases} \min_{|x|=|y|-1} \text{Support}(x), & |Y| = 3 \\ \min_{|x|=|y|-1} \text{Support}^+(x), & |Y| > 3 \end{cases}$$

Cette étape est répétée jusqu'à ce qu'aucun candidat ne puisse être généré. On signale ici qu'on n'a pas besoin de diffusés les k-Itemsets candidats générés au niveau de chaque site

Maître d'un groupe puisque tous les k-Itemsets candidats générés sont identiques dans tous les sites Maîtres.

- **Etape 3 : raffinage des Itemsets fréquents :**

Une fois le calcul des k-Itemsets fréquents globaux est terminé, la phase de raffinage est lancée:

- Chaque site Maître de chaque groupe envois aux sites Esclaves de son propre groupe la liste des k-Itemsets candidats globaux construite dans l'étape précédente (tous les sites Maîtres ont les mêmes candidats).
- Les sites Esclaves du groupe calculent les supports locaux des k-Itemsets reçus en parcourant leurs portions locales de base de données et ils renvoient les résultats au site Maître du groupe.
- Les sites Maîtres des différents groupes diffusent (broadcaste) entre eux les valeurs des supports globaux exactes des k-Itemsets globaux pour déterminer tous les k-Itemsets fréquents globaux de tous les sites participants de la Grille. On signale que la liste des 1-Itemsets et des 2-Itemsets fréquents globaux exactes n'est pas incluse dans cette diffusion car ils sont calculés dans l'étape1.

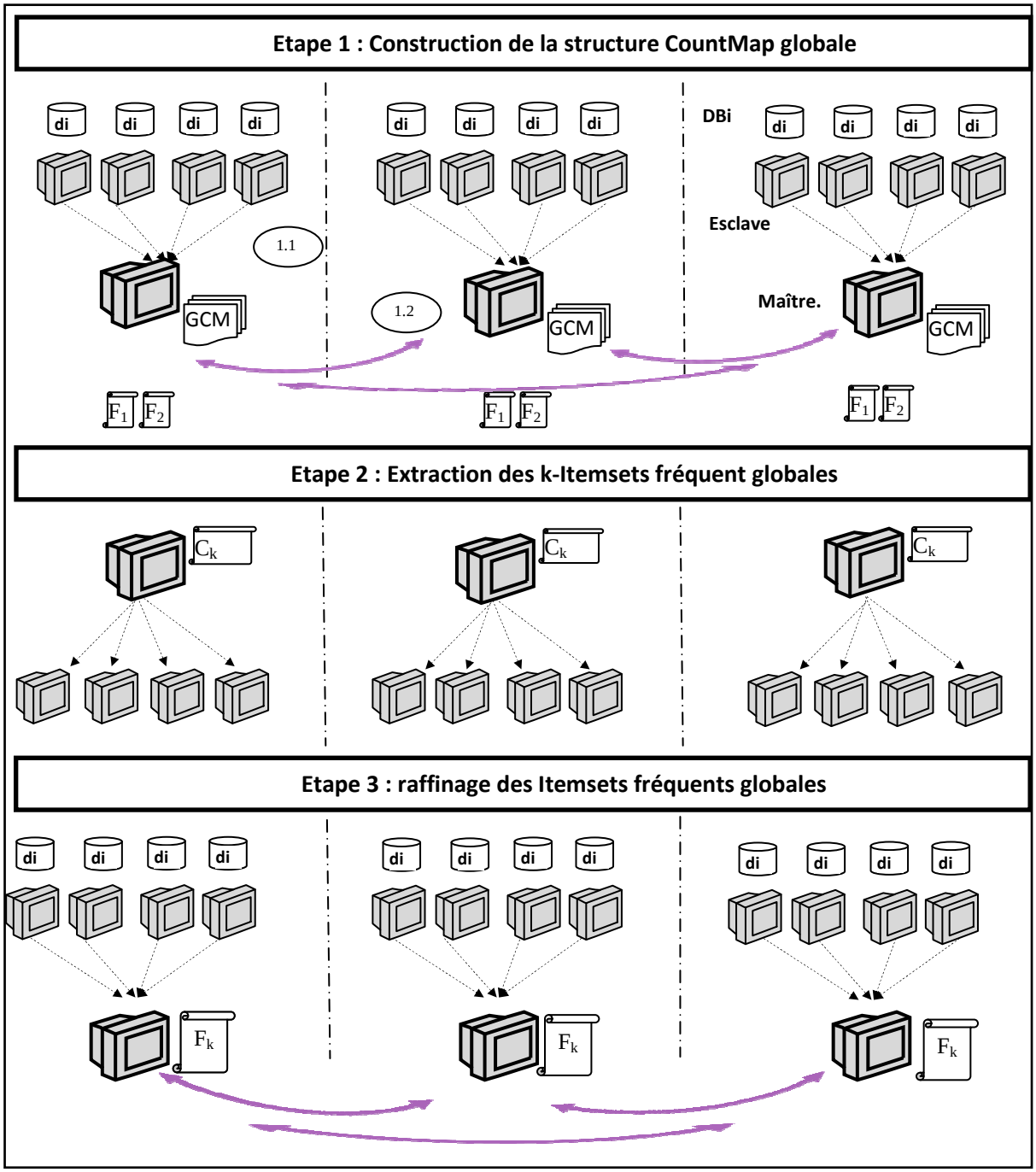
```
1.  LCM(Si)= Gen-CountMap(D(i) ,I) ;          //  construction de la structure CountMap
2.  Send(S(i),M(j), LCM(Si)) ;                //     local au niveau des sites Esclaves
3.  Pour chaque Site Maitre j faire
4.  | GCM(j)=∑ LCM(Si) ;
5.  |  Pour chaque Site Maitre k≠j faire      // construction de la structure CountMap
6.  | |  Send(M(j), M(k), GCM(j)) ;          // global au niveau du groupe j
7.  |  fin ;
8.  | GCM=∑ GCM(i) ;
9.  fin ;
10. Pour chaque Site Maitre j faire
11. |  GL1 =GetFrequentItemsets(GCM,1) ;
12. | GL2 =GetFrequentItemsets(GCM,2) ;
13. |  GL2+ = GL2 ;
14. |  Pour  (k= 3; GLk-1+≠Ø; k++) faire
15. | |  GLk+ = CM-gen(GLk-1+);              // génération des Itemsets Candidats
16. |  fin;
17. |  CGL+ ={∪ GLk+ |k ≥ 3} ;
18. |  Pour chaque Site Esclave i faire
19. | |  Send(M(j), S(i), CGL+ ) ;           //  calcul des supports des
20. | |  Pour chaque transaction t ∈ D(i)  faire   //  Itemsets candidats
21. | | | Ct(i)= sous-ensemble (CGL+ ,t);    //  au niveau du group j
22. | | |  Pour chaque candidates c ∈ Ct(i) faire
23. | | | |  c.freq++;
24. | | |  fin;
25. | |  fin;
26. | |  Send(S(i),M(j), CGL+ ) ;
27. |  fin ;
28. |  Pour chaque Site Maitre k≠j faire
29. | |  Send(M(j), M(k), CGL+ ) ;           // diffusion des resultats entres les Maitres
30. |  fin ;
31. | GLk={c∈ CGL+ |c.sup ≥ minsup}          // le site Maitre détermine les k-Itemset fréquenst globale k≥3
32.  FIN ;
33. Résultats= ∪k GLk
```

Fig 3.8 Algorithme AprioriGCM

Nous remarquons que cet algorithme est bien adapté à l'exécution sur des plateformes géographiquement distribuées à grande échelle comme les Grilles.

L'algorithme permet d'assurer d'une certaine scalabilité à grande échelle et cela en segmentant les sites en groupes de calculs et en déléguant la coordination de ces sites aux sites Maitres.

De plus, cette segmentation permet d'assurer un certain niveau d'équilibre de charge en confiant aux sites dits Esclaves la tache de calcul des supports d'Itemsets et de l'accès aux sources de données et aux sites Maitres la tâche de la diffusion des résultats.


Etape 1 : Construction de la structure CountMap globale
di
1.1
GCM
1.2
DBi
Esclave
Maître.
F1
F2
Etape 2 : Extraction des k-Itemsets fréquent globales
Ck
Etape 3 : raffinage des Itemsets fréquents globales
Fk

5.2 Déploiement de la méthode sur les Grilles

La vision actuelle pour le DataMining distribuées sur les plateformes Grilles implique souvent les services web et les standards tels que le WSRF. Les services web sont un sujet actif dans le domaine de calcul distribué. L'idée générale est de fournir des services de données et de traitement sous forme d'un service web générique que l'utilisateur d'intérêt.

Les Grilles vont au-delà du modèle de service du Web et fournissent des services d'ordonnancements, de gestion des ressources, de réservation de stockage, d'assurance de qualité de service, de monitoring et d'autres services. Ces services fournissent un meilleur modèle d'organisation des ressources partagées qui permet une utilisation efficace des ressources.

Nous avons donc exploré les possibilités d'exposer notre méthode d'extraction des règles d'associations basée sur l'algorithme AprioriGCM sous forme de service Grille.

L'implémentation et le teste de notre algorithme sur une plateforme Grille n'est pas encore réalisé. Cette tâche nécessite un savoir faire et une certaine connaissance de la technicité des environnements Grilles. Nous avons donc se restreindre à présenter l'aspect générale de l'algorithme et à laisser la tâche d'implémentation aux travaux à venir.

Nous proposons d'exposer notre méthode sous forme d'un service Grille *GCMS* (Grid-enabled AprioriGCM service) qui est composé de deux services Grilles, à savoir Un service de calcul de CountMap et un service d'extraction des Itemsets fréquents.

Ainsi, l'étape 1 de l'algorithme AprioriGCM qui concerne le calcul de la structure de données CountMap globale est assurée par le **service CountMap.** Les étapes 2 et 3 qui concernent l'extraction des Itemsets fréquents globaux ainsi que leurs raffinages sont assurés par le **service d'Extraction Des Itemsets.** La figure suivante présente un cas pratique d'utilisation de la méthode dans un environnement Grille :

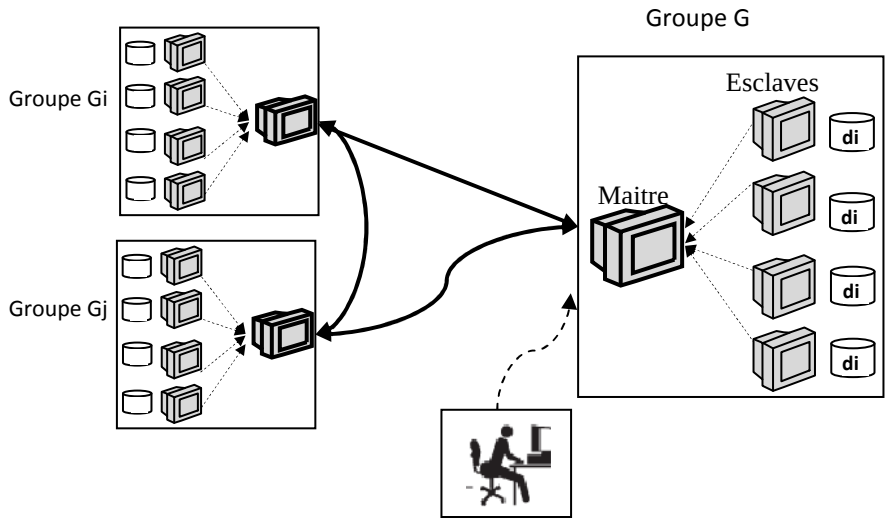


Fig. 3.9 Un scénario typique d'utilisation de la méthode AprioriGCM

La figure présente une organisation virtuelle d'une Grille regroupant plusieurs groupes/domaines administratifs. Un utilisateur potentiel invoque le service GCMS de son groupe en envoyant les paramètres Minsup, Minconf. Cette requête sera renvoyée à l'instance du service GCMS du groupe d'appartenance du client. Le site Maître diffuse cette requête au autres Maîtres. Cette requête va créer des instances de service CountMap dans chaque groupe.

Le service CountMap va lancer le calcul de la structure CountMap globale comme on a présenté dans l'étape 1 de l'algorithme AprioriGCM : le site Maître construit la structure CountMap globale de son groupe et la diffuse aux autres maîtres pour le calcul de la structure CountMap globale de tous les groupes.

Une fois le service CountMap est achevé, le service d'extraction d'Itemsets est lancé par les sites Maîtres. Chaque instance du service crée l'ensemble des Itemsets candidats en se basant sur le $Support^{+}$ des Itemsets (étape2) et suivi par l'étape3 de raffinage des résultats comme la présenté dans l'algorithme AprioriGCM.

Certaines améliorations peuvent être introduites pour améliorer l'utilisation de notre méthode ;

5.2.1 Mise en cache de la structure CountMap :

Nous avons remarqué que l'étape 1 de construction de la structure CountMap peut être mise en cache par le site Maître pour son exploitation ultérieure. Une fois le service CountMap terminé, la structure CountMap sera enregistrée par le site Maître. Les prochaines requêtes des clients seront interceptées par le service GCMS mais cette fois-ci, nous démarrons directement le service d'extraction des Itemsets. Cela nous permet d'éviter le recalcule de la structure CountMap à chaque requête d'un client.

5.2.2 Utilisation de la version hybride pour le service d'extraction des Itemsets.

Une autre amélioration que nous envisageons est l'utilisation d'hybridation entre l'utilisation de la phase de validation à chaque itération et l'utilisation de la phase de validation à la fin de l'extraction des Itemsets.

La méthode consiste à introduire au niveau de service d'extraction des Itemsets fréquents un paramètre $S_{Candidats}$ qui représente un seuil de nombre maximal de candidats générés sans validation. L'idée est de générer les Itemsets fréquents par l'utilisation du $Support^{+}$ sans la validation des résultats, une fois le seuil $S_{Candidats}$ est atteint, le service d'extraction d'Itemset fréquent lance la phase de validation pour raffiner l'ensemble des Itemsets fréquents calculés jusqu'à maintenant. A la fin de cette validation, le service reprend de nouveau l'opération de génération d'Itemsets fréquents par l'utilisation du $Support^{+}$ sans la phase de validation. Le service d'extraction d'Itemsets fréquents répète ces deux opérations jusqu'à la fin du calcul de tous les Itemsets fréquents.

Avec cette méthode, on garde les avantages des deux versions de l'algorithme distribué AprioriDCM à savoir, la réduction du nombre de communication/synchronisation entres les sites et la disponibilité des résultats valides.

La performance de cette méthode dépend du choix de la valeur du seuil $S_{Candidats.}$ Le chois d'une valeur petite permet de lancé la validation de l'ensemble d'Itemsets fréquents le plus souvent possible et donc un nombre important d'échange entre les sites Esclaves et les sites Maîtres ; le choix d'une valeur élevée nous permet de réduire le nombre d'échange entre les sites avec une phase de validation plus importante (un nombre important d'Itemsets parasites).

Le seuil $S_{Candidats}$ est le paramètre clé dans cette situation. Sa valeur doit être bien étudiée pour garantir une meilleure performance de la méthode.

6 Conclusion

Les Grilles sont utilisées comme une plateforme pour l'implémentation et le déploiement d'applications et de services de gestion et d'extractions des connaissances géographiquement distribuées. Les travaux de recherche montrent comment l'utilisation des technologies Grilles peut apporter plusieurs avantages pour l'implémentation d'applications distribuées d'extraction des connaissances.

L'algorithme AprioriGCM est une adaptation de notre algorithme AprioriDCM pour une exécution sur infrastructure distribuée de type Grille. Les performances d'AprioriDCM sont limitées au regard à l'utilisation d'un nombre important de nœuds. AprioriGCM repousse ces limites et permettant l'utilisation de la méthode à grande échelle.

Nous avons aussi étudié la possibilité d'utilisation des technologies des grilles au Grilles par la présentation de la méthode sous forme de services Grilles.

REFERENCES BIBLIOGRAPHIQUES

[AGR 93] AGRAWAL (R.), IMIELINSKI (T.), SWAMI (A.). "*Mining Association Rules between sets of items in large Databases*". Proceedings of the ACM SIGMOD Intl. Conference on Management of Data, Washington, USA, June 1993, p. 207 - 216.

[AGR 94] AGRAWAL (R.), SRIKANT (R.), "*Fast Algorithms for Mining Association Rules in Large Databases*". Proc. of the 20th Int'l Conf. on Very Large Data Bases (VLDB), juin 1994, p. 478 - 499, version étendue : IBM Research Report RJ 9839.

[AGR 96] Agrawal (R.), Shafer(J. C.),"*Parallel Mining of Association Rules*". IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 962-969, December 1996.

[ALL 01] Allcock (W.), Bester (J.), Bresnahan (J.), Chervenak (A.), Liming (L.), Tuecke (S.). "*GridFTP: Protocol Extensions to FTP for the Grid"*, (2001). Sur: http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFCDraft.pdf

[ALI 05] Ali (A.S.), Rena (O.), Taylor (I.). "*Web services composition for distributed data mining*". In 'Proc of the 2005 IEEE International Conference on Parallel Processing Workshops, ICPPW'05, 2005.

[ASH 04] Ashrafi (M. Z.), Taniar (D.), Smith (K.). "*ODAM: An Optimized Distributed Association Rule Mining Algorithm*". IEEE Computer Society, IEEE Distributed Systems Online 1541-4922 © 2004, Vol. 5, No. 3; March 2004

[BER 96] BERRY (M.J.A.), LINOFF (G.). "*Data Mining-Techniques appliquées au marketing, à la vente et aux services clients*" - 1996 MASSON (Wiley).

[BRE 05a] Brezany (P.), Kloner (C.), Tjoa (A. M.). "*Development of a grid service for scalable decision tree construction from grid databases*", In 'Sixth International Conference on Parallel Processing and Applied Mathematics', Poznan. (2005).

[BRE 05b] Brezany (P.), Janciak (I.), Tioa (A. M.). "*GridMiner: A fundamental infrastructure for building intelligent grid systems*". In 'The 2005 IEEE/WIC/ACM International Conference of Web Intelligence, WI'05, 2005.

[BRE 06] Brezany (P.), Janciak (I.), Han (Y.). "*Parallel and distributed grid services for building classification models based on neural networks*". In 'Second Austrian Grid Symposium', Innsbruck. (2006)

[BRI 97] Brin (S.), Motwani (R.), Ullman (J. D.), Tsur (S.). "*Dynamic itemset counting and implication rules for market basket data*". In Proceedings ACM SIGMOD, International Conference on Management of Data, Tucson, Arizona, USA. ACM Press pp. 255 - 264, 1997.

[BUE 04] Bueti (G.), Congiusta (A.), Talia (D.), "*Developing distributed data mining applications in the Knowledge Grid framework*". In Int. Conf. on High-Performance Computing for Computational Science (VECPAR'04)', Vol. 3402 of LNCS, Springer, Valencia, pp. 156–169, (2004).

[CAN 03] Cannataro (M.), Talia (D.). "*The Knowledge Grid*", Communications of the ACM 46 (1), 89–93. (2003).

[CAN 04a] Cannataro (M.), Comito (C.), Congiusta (A.), Veltri, (P.). "*PROTEUS: a bioinformatics problem solving environment on grids*". Parallel Processing Letters 14 (2), 217–237, (2004)

[CAN 04b] Cannataro (M.), Congiusta (A.), Pugliese (A.), Talia (D.), Trunfio (P.), "*Distributed data mining on grids: services, tools, and applications*", IEEE Transactions on Systems, Man, and Cybernetics: Part B 34 (6), 2451–2465, (2004)

[CHA 97] Chattratichat (J.), Darlington (J.), Ghanem (M.). "*Large Scale Data Mining: Challenges and Responses"*, Proceedings of the 3th International Conference on Knowledge Discovery and Data Mining, pp. 143-146, August 1997.

[CHA 99] CHAPIN (S.), KARPOVICH (J.), GRIMSHAW (A.). "*The legion resource management system*". In Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing. IEEE Press, Los Alamitos, CA. (1999).

[CHE 96a] Cheung (D. W.), Han (J.), Vincent (Ng.), Fu (A. W.), Fu (Y.). "*A Fast Distributed Algorithm for Mining Association Rules*". Proceedings of PDIS, 1996.

[CHE 96b] Cheung(D. W.), Vincent (T. Ng), Fu(A. W.), Fu(Y.), "*Efficient Mining of Association Rules in Distributed Databases*", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 911-922, December 1996.

[CHE 98] Cheung(D.), Xiao (Y.). "*Effect of Data Skewness in Parallel Mining of Association Rules*". Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, Vol. 1394, Springer-Verlag, New York, 1998, pp. 48–60.

[CNA 98] Conservatoire National des Arts et métier de Lille "*Data warehouse et Data Mining*". version 1.1, le 15 juin 1998, disponible sur : www.imi-utc.com/ptemoign/index.html

[CON 07] Congiusta (A.), Talia (D.), Trunfio (P.). "*Distributed data mining services leveraging WSRF*", Future Generation Computer Systems 23 (1), 34–41. (2007)

[CRA 05] Craus (M.), Aflori (C.). "*Association Rules Discovery using Grid Services*". In '3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications', March 27-31, 2005 – TUNISIA.

[CZA 04] Czajkowski(K.), Ferguson (D. F.), Foster (I.), Frey (J.), Graham (S.), Sedukhin (I.), Snelling (D.), Tuecke (S.), Vambenepe (W.). "*The WS-Resource Framework version 1.0*", 2004. Disponible sur: http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf.

[DUQ 99] Duquenne (V.). "*Latticial Structures in Data Analysis*", TCS, vol. 217, no 2, 1999, p. 407 - 436.

[ELS 05] Elsayed (I.), Brezany (P.). "Online analytical mining of association rules on the grid", Technical Report Deliverable of the TU/Uni-Vienna GridMiner Project, Institute for Software Science, University of Vienna. (2005).

[ERW 01] Erwin (D. W.), Snelling (D. F.). "*UNICORE: a grid computing environment*". In Int. Conf. on Parallel and Distributed Computing (Euro-Par'01)', Vol. 2150 of LNCS, Springer, Manchester, UK, pp. 825–834. (2001)

[ERW 04] Erwin (D. W.), Snelling (D. F.). "*Middleware for the next generation grid infrastructure*". In 'International Conference on Computing in High Energy and Nuclear Physics (CHEP'04)', Interlaken, 2004.

[FIO 06] FIOLET (V.). "*Algorithmes distribues d'extraction de connaissances*" . Thèse Doctorat, Université des sciences et technologies de LILLE, (2006)

[FOS 95] Foster (I.), Geisler (J.), Nickles (W.), Smith (W.), Tuecke (S.). "*Software infrastructure for the I-WAY high performance distributed computing experiment*". In Proc. 5th IEEE Symposium on High Performance Distributed Computing (1995)

[FOS 97] Foster (I.), Kesselman (C.). "*Globus: A metacomputing infrastructure toolkit*". International Journal of Supercomputer Applications and High Performance Computing 1997; 11(2):115–128.

[FOS 98] Foster (I.), Kesselman (C.). "*The Globus Project: A Status Report*". In Proc. Heterogeneous Computing Workshop, IEEE Press, 1998, 4-18.

[FOS 01] Foster (I.), Kesselman(C.), Tuecke (S.). "*The anatomy of the grid: Enabling scalable virtual organizations*". International J. Supercomputer Applications, 15(3), (2001).

[FOS 02] Foster (I.), Kesselman (C.), Nick (J.), Tuecke (S.). "*The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*", tech. report, Globus Project; 2002. Disponible sur : http://www.globus.org/alliance/publications/papers/ogsa.pdf

[FOS 03] Foster (I.), Kesselman (C.). "*Computational grids*". In 'The grid: Blueprint for a New Computing Infrastructure', Eds. Morgan Kaufmann, 2003

[FOS 05] Foster (I.). "*Globus Toolkit Version 4: Software for Service-Oriented Systems*", IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005

[GGF] Global Grid Forum, en ligne sur : http://www.globus.org/

[GHA 02] Ghanem (M.), Guo (Y.), Lodhi (H.), Zhang (Y.), "*Automatic scientific text classification using local patterns*": KDD Cup 2002 (Task 1)', SIGKDD Explorer Newsletter 4 (2), 95–96. (2002).

[GHA 05] Ghanem (M.), Ratcliffe (J.), Curcin (V.), Li (X.), Tattoud (R.), Scott (J.), Guo (Y.). "*Using text mining for understanding insulin signaling*". In '4th UK e-Science All Hands Meeting 2005'. (2005).

[GLO] Globus Project. http://www.globus.org.

[GRA 06] Graham (S.), Karmarkar (A.), Mischkinsky (J.), Robinson (I.), Sedukhin (I.), "*Web Service Resource Framework"*. Disponible sur : http://docs.oasis-open.org/wsrf/wsrf-ws resource-1.2-spec-pr-01.pdf

[GUO 05] Guo (Y.), Liu (J.G.), Ghanem (M.), Mish (K.), Curcin (V.), Haselwimmer (C.), Sotiriou (D.), Muraleetharan (K. K), Taylor (L.). "*Bridging the macro and micro: a computing intensive earthquake study using Discovery Net*", in 'Proceedings of the 2005 ACM/IEEE conference on Supercomputing (SC'05)', IEEE Computer Society, Washington, DC, p. 68. (2005).

[JAN 06] Janciak (I.), Sarnovsky (M.), Tjoa (A. M.), Brezany (P.). "*Distributed classification of textual documents on the grid*". In 'The 2006 International Conference on High Performance Computing and Communications, LNCS 4208", Munich, pp. 710–718. (2006).

[KAR 00] Kargupta (H.), Kamath (C.), Chan (P.). "*Distributed and parallel data mining: Emergence, growth, and future directions*". In 'Advances in Distributed and Parallel Knowledge Discovery'(2000), AAAI/MIT Press, pp. 409–416

[KHO 04] Khoussainov (R.), Zuo (X.) and Kushmerick (N.). "*Grid-enabled Weka: a toolkit for machine learning on the grid*", ERCIM News. (2004).

[LUO 06] Luo (C.), Pereira (A.L.), (S.M.) Chung. "*Distributed Mining of Maximal Frequent Itemsets on a Data Grid System*". The Journal of Supercomputing, 37, 71–90, 2006

[LUQ 06] Lu (Q.), Hao (P.), Curcin (V.), He (W.), Li (Y.), Luo (Q.)., Guo (Y.),.Li (Y.)."*KDE Bioscience: platform for bioinformatics analysis workflows*", Journal of Biomedical Informatics 39 (4), 440–450. (2006).

[Pas 00] PASQUIER (N.). "*Extraction de Bases pour les Règles d'Association à partir des Itemsets Fermés Fréquents*", Laboratoire d'Informatique (LIMOS) - Université Clermont-Ferrand II, janvier 2000

[PIA 91] Piatetsky-Shapiro (G.), Frawely (W.J.). "*Knowledge Discovery in Databases*". AAA Press, 1991.

[PER05] Pérez (M. S.), Sanchez (A.), Herrero (P.), Robles (V.), Pena (J. M.). "*Adapting the Weka data mining toolkit to a grid-based environment*", in 'AWIC, Lecture Notes in Computer Science 3528', pp. 492–497. (2005).

[PER 07] Pérez (M. S.), Sanchez (A.), Robles (V.), Herrero (P.), Pena (J.M.), "*Design and implementation of a data mining grid-aware architecture*", Future Generation Computer Systems 23 (1), 42–47, (2007).

[PAR 95] Park(J. S.), Chen (M.), Yu(P. S.), "*An Effective Hash Based Algorithm for Mining Association Rules*", Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 175-186, San Jose, California, 22-25 May 1995.

[ROW 03] Rowe (A.), Kalaitzopoulos (D.), Osmond (M.), Ghanem (M.), Guo (Y.). '*The Discovery Net system for high throughput bioinformatics*", Bioinformatics 19 (1), i225–i231. (2003)

[RUS 05] Rushing (J.), Ramachandran (R.), Nair (U.), Graves (S.),Welch (R.), Lin (A.), "*ADaM: a datamining toolkit for scientists and engineers*", Computers and Geosciences 31, 607–618. (2005).

[SAB 04] SABATIER (F.). Memoire en informatique : "*Conception et déploiement d'une Grille de contrôle de processus physiques »*. Conservatoire National des arts et Métiers, Centre Régional De Lorraine, Centre d'enseignement de Metz, 6 Septembre 2004

[SAI 03] Sairafi (S.A), Emmanouil (F.S.), Ghanem (M.), Giannadakis (N.), Guo (Y.), Kalaitzopolous (D.), Osmond (M.), Rowe (A.), Syed (J.), Wendel (P.), "*The design of discovery net: Towards open grid services for knowledge discovery*", International Journal of High Performance Computing Applications 17, 2003.

[SAK 08] Sakthi (U.), Hemalatha (R.), Bhuvaneswaran (R. S.). "*Parallel and Distributed Mining of Association Rule on Knowledge Grid."*. In "World Academy of Science, Engineering and Technology 42", (2008)

[SALL 03] Salleb (A.). Thèse doctorat : "*Recherche de motifs fréquents pour l'extraction de règles d'association et de caractérisation*". Université d'Orléans, décembre 2003.

[SAN 04] Sanchez (A.), Sanchez (J. M. P.), Pérez (M. S.), Robles (V.), Herrero (P.). "*Improving distributed data mining techniques by means of a grid infrastructure*", in 'OTMWorkshops. LNCS 3292', pp. 111–122. (2004)

[SAN 08] Sanchez (A.), Perez (M. S.), Gueant (P.), Pena (J. M.), Herrero (P.). "*DMGA: A generic brokering-based Data Mining Grid Architecture*". In 'Data Mining Techniques in Grid Computing Environments' Edited by Werner Dubitzky, John Wiley & Sons, Ltd (2008).

[SET] Projet SETI@home. http://setiathome.ssl.berkeley.edu/.

[SCH 01] Schuster (A.), Wolff (R.). "*Communication-efficient distributed mining of association rules*", In Proc. of the 2001 ACM SIGMOD Int'l. Conference on Management of Data, Santa Barbara, California, pp. 473 ñ 484. (2001)

[SHA 96] Shafer (J. C.), Agrawal (R.), Mehta (M.). "*SPRINT: A scalable parallel classifier for data mining*". In Proceedings of the 22nd International Conference on Very Large Databases (VLDB'96)', Morgan Kaufmann, pp. 544–555, 1996.

[SHI 96] Shintani (T.), Kitsuregawa (M.), "*Hash Based Parallel Algorithms for Mining Association Rules*". Proc. 4th Int. Conf. Parallel and Distributed Information Systems, IEEE Computer Soc. Press, Los Alamitos, Calif., 1996, pp. 19–30.

[TAL 06] Talia (D.). "*Grid-based Distributed Data Mining Systems, Algorithms and Services"*. In HPDM 2006: 9th International Workshop on High Performance and Distributed Mining. Bethesda, MD, USA, (2006).

[TAY 07] Taylor (I.), Shields (M.), Wang (I.), Harrison (A.). "*The Triana workflow environment: architecture and applications"*. In 'Workflows for e-Science', Springer, New York, pp. 320–339. (2007).

[TUE 03] Tuecke (S.), Czajkowski (K.), Foster (I.), Frey (J.), Grahame (S.), Kesselman (C.), Maguire (T.), Sandholm (T.), Venderbilt (P.), Snelling (D.). "*Open Grid Services Infrastructure (OGSI) version 1.0*". Global Grid Forum Draft Recommendation, June 27, 2003.

[UNI 03] Unicore Forum. Unicore Plus Final Report*: "Uniform Interface to Computing Resource"*. 2003. Disponible sur: http://www.unicore.org/documents/UNICOREPlus-Final-Report.pdf.

[W3C 00] World Wide Web Consortium (W3C). "*Simple Object Access Protocol (SOAP) 1.1"*. W3C. Note 08 May 2000. Sur: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[W3C 02] W3C, Web Services, 2002, http://www.w3.org/2002/ws/

[WEG 07] Wegener (D.), May (M.). "*Extensibility of grid-enabled data mining platforms: a case study*", in 'Proceedings of the 5th International Workshop on Data Mining Standards, Services and Platforms (KDD 2007)', San Jose, USA, pp. 13–22. (2007).

[WIT 02] Witten (I. H.), Frank (E.), "*Data mining: practical machine learning tools and techniques with Java implementations*', SIGMOD Record 31 (1), 76–77. (2002)

[WIT 05] Witten (I.), Frank (E.). "Practical Machine Learning Tools and Techniques". 2nd edn, Morgan Kaufmann,(2005).

[ZAK 96] Zaki (M. J.), Ogihara(M.), Parthasarathy (S.), Li (W.). "*Parallel Data Mining for Association Rules on Shared-Memory Multiprocessors*". Technical Report TR 618, University of Rochester, Computer Science Department, May 1996.

[ZAK 97] Zaki (M. J.), Parthasarathy (S.), Ogihara (M.), Li (W.). "*New algorithms for fast discovery of association rules*". In 3rd Intl. Conf. on Knowledge Discovery and Data Mining. AAAI Press pp. 283 - 296, 1997.

TRAVAUX REALISES

[MOU 10a] MOUNA (A.), BELBACHIR (H.). "*Distributed Association Rules On Grids*". Intl. Conference on Information and Knowledge Engineering, Singapore, SINGAPORE, August 25-27,2010.

[MOU 10b] MOUNA (A.), BELBACHIR (H.). "*Distributed Association Rules On Grids*". Intl. Conference on Computer, Electrical, and Systems Science, and Engineering, Amsterdam, NETHERLANDS, September 28-30,2010.

Conclusions et perspectives

Nous avons étudié à travers ce document le problème d'extraction des règles d'association dans un contexte distribué et en particulier dans les Grilles. Les Grilles sont devenues des infrastructures de calcul distribué bien accepté dans les sciences et les industries.

L'accent à été mis sur l'étape la plus coûteuse du processus d'extraction des règles d'association à savoir l'extraction des Itemsets fréquents.

Notre objectif était d'améliorer les performances de cette étape en proposant de nouvelles solutions qui tiennent en comptes des spécificités des supports d'exécutions particulièrement les Grilles.

Sur la base d'une étude approfondie des algorithmes distribués d'extractions des Itemsets fréquents, nous avons proposé une nouvelle stratégie pour le problème d'extraction d'Itemsets fréquents en contexte distribué qui est basée sur la notion de Support^{+}.

Le schéma général consiste à utiliser des déductions sur les valeurs des supports d'Itemsets pour l'extraction des Itemsets fréquents et l'utilisation des stratégies de communications plus scalables et plus adaptées aux contextes d'exécution comme les Grilles.

Notre approche permet de réduire le nombre d'Itemsets candidats et/ou le nombre de parcours de la base de données. D'autre part, elle réduit le coût des communications/synchronisations nécessaires pour l'échange d'Itemsets candidats et/ou le coût de calcul des supports locaux de ces candidats.

Les expérimentations effectuées ont permis de valider notre approche et de prouver son utilité dans l'amélioration des performances de l'étape d'extraction d'Itemsets fréquents en contextes distribués.

Plusieurs pistes ont été abordées pour l'adaptation de notre approche à une utilisation dans un environnement Grilles à savoir :

- La présentation d'un scénario typique de déploiement de notre méthode sous forme d'un service Grille
- L'utilisation d'une stratégie de communication plus scalable et qui reflète l'architecture réseau réels des environnements Grilles (organisations virtuelles).
- La réduction du nombre de synchronisations par l'utilisation d'une version qui nécessite seulement deux parcours de la base de données.

De nombreuses perspectives sont envisagées pour la suite de ce travail, à savoir :

- **Amélioration de notre méthode par l'utilisation des représentations condensées telle que les Itemsets fermés fréquents** : notre méthode est basée sur l'algorithme générique Apriori. Or, les Itemsets fermés fréquents permettent de réduire le nombre de candidats à considérer et semblent intéressants pour l'amélioration des performances de notre approche.
- **Déploiement de notre méthode sur une Grille réelle** : nous pensons que le déploiement de notre algorithme sur un environnement Grille réel nous permet d'aborder de nouveaux problèmes spécifiques à l'utilisation de la technique des règles d'association dans les Grilles à savoir : la sécurité, l'accès aux sources de données, les outils de visualisations, etc.
- **Investigation des algorithmes de générations des règles d'association** : cette étape postérieure à l'extraction d'Itemsets fréquents n'a pas eu beaucoup d'attentions comme l'étape d'extraction d'Itemsets fréquents. Cependant, le nombre de règles générées et leurs redondances posent de vrais problèmes lorsque leur nombre atteints des centaines ou des milliers de règles. Nous abordons dans les travaux futurs ce problème en étudiant les différentes solutions existantes à savoir les règles d'associations informatives et les possibilités d'aborder ce problème d'une façon distribuée.

Résumé

Les techniques de DataMining particulièrement l'extraction de règles d'associations permettent de découvrir des connaissances qui servent à l'aide à la décision. Nous avons proposé une nouvelle starégie pour le problème de l'extraction distribuée des règles d'association sur les Grilles en particulier dans la phase d'extraction d'itemsets fréquents.

Notre approche consiste à modifier la phase de génération-élaguage des itemsets candidats par l'introduction d'une nouvelle méthode qui se base sur l'utilisations des déductions sur les valeurs des supports d'itemsets pour l'extraction des itemsets fréquents et sur la proposition des stratégies plus scalables et plus adaptées de l'utilisation de notre méthode dans les environnements distribués en général et sur les environnements Grilles en particulier.

Notre approche permis de réduire le nombre d'itemsets candidats et/ou le nombre de parcours de la base de données d'une part et le cout de communications/synchronisations nécessaire pour l'échange de ces candidats et/ou le calcul des supports locaux de ces candidats entre les différents sites géographiquement distribués d'une autre part.

Les éxpérimentations éffféctuées ont permet de valider notre approche et de prouver son utilité dans l'amélioration des performances de l'étape d'extraction d'itemsets fréqunets en contextes distribués.

Mots-Clés:

DataMining Haut performance, règles d'association distrubées, l'extraction des itemets fréquents, Grilles de données.

Abstract

DataMining techniques especially association rules allow to discover knowledges which help decision makers. We proposed a new strategy for the problem of distributed association rules on Grids particularly on the frequent Itemset mining step.

Our approach consist in the modification of generation-pruning candidates Itemsets stage by introducing a new method based on the use of deductions on Itemsets support values for the frequent Itemsets mining and on the proposition of strategies more scalable and more suitable to the use of our method on general distributed frameworks and on Grids.

Our approach allows on one hand to reduce the candidate Itemsets number and/or the database scans number and on the other hand to reduce the communications/synchronizations cost required for the exchange of this candidate Itemsets and/or for the calculation of the locales counts of these candidates in the different geographically distributed sites.

The experiments made have allowed us to validate our approach and to prove it usefulness on improving the performances of the frequent Itemsets mining step on distributed contexts.

Key-words:

High performance Data mining, distributed association rules, Grids-based frequent Itemsets mining, Data Grid.