Echantillonnage optimisé de données temporelles distribuées pour l'alimentation des entrepôts de données

Raja Chiky*, Georges Hébrail*,**

* GET-ENST Paris
Laboratoire LTCI - UMR 5141 CNRS - Département Informatique et Réseaux
46 rue Barrault, 75634 Paris Cedex 13
Email: prenom.nom@enst.fr
** EDF R&D - Département ICAME
1, Avenue du Général de Gaulle, 92140 Clamart
Email: georges.hebrail@edf.fr

Résumé. Les entrepôts de données sont de plus en plus alimentés par des données provenant d'un grand nombre de capteurs. Les capteurs trouvent leur utilité dans plusieurs domaines : médical, militaire, trafic routier, météorologie ou encore des données de consommation électrique. Pour faire face à la volumétrie et au taux d'arrivée des flux de données, des traitements sont effectués à la volée sur les flux avant leur enregistrement dans les entrepôts de données. Nous présentons des algorithmes d'échantillonnage optimisé sur des flux de données provenant de capteurs distribués. L'efficacité des algorithmes proposés a été testée sur un jeu de données de consommation électrique.

1 Introduction

Les entrepôts de données (data warehouses) sont utilisés afin d'améliorer la prise de décision dans les entreprises. Ils servent à historiser des données résumées, non volatiles et disponibles pour l'interrogation, l'analyse et la prise de décision. Les entrepôts de données sont de plus en plus alimentés par des données provenant d'un grand nombre de capteurs distribués. On retrouve ces capteurs dans des domaines aussi divers que la météorologie (établir des prévisions), le domaine militaire (surveiller des zones sensibles), l'analyse des consommations électriques (transmettre des alertes en cas de consommation anormale),... Le concepteur d'un entrepôt de données doit mettre en place une stratégie de mise à jour pour l'historisation en prenant en compte la volumétrie des données et en garantissant les meilleures performances possibles pour l'entrepôt en terme de temps de réponse pour l'interrogation et l'analyse.
Cet article traite des problèmes liés à des données temporelles et distribuées, mesurées en temps réel, où les sources des données correspondent à un grand nombre de capteurs qui enregistrent périodiquement des mesures dans des domaines spécifiques (température, consommation électrique...). La figure 1 montre un exemple d'architecture de récupération de données à partir de capteurs servant à mesurer des index de consommation électrique. Il s'agit de plusieurs millions de compteurs électriques communicants qui sont reliés à des concentrateurs, qui à leur tour sont reliés à des entrepôts de données. L'envoi des données se fait sous forme de flux,

arrivant de façon continue à un pas de temps très fin. Ceci génère une quantité volumineuse de données, et il devient coûteux de stocker et traiter toutes ces données. De plus, le concentrateur récoltant les données impose une limite de bande passante à ne pas dépasser et les données relevées à chaque pas de temps ne doivent pas excéder cette limite.
Le coût élevé de la collecte de données et la contrainte de la bande passante nous incitent à

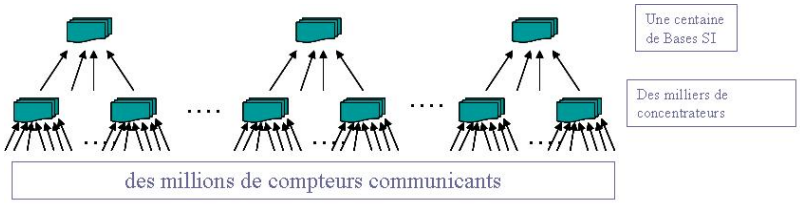

FIG. 1 – *Structure hiérarchique de collecte de données*

sélectionner un ensemble minimal de données échantillonnées tout en s'assurant de la représentativité de ces données pour s'approcher du flux de données initial. Le sur-échantillonnage est très coûteux en temps et espace de stockage. D'autre part, le sous-échantillonnage mène à une évaluation imprécise des données.
L'échantillonnage en ligne a été largement étudié dans le cadre des flux de données (Muthukrishnan, 2005; Golab et al., 2003; Babcock et al, 2002). Toutefois, la plupart des travaux se concentrent sur l'échantillonnage d'un unique flux de données et n'exploitent pas des possibilités de communication entre un concentrateur et ses sources de données. Nous présentons dans cet article deux algorithmes d'échantillonnage en ligne appliqués à des flux de données fortement distribuées. Ces algorithmes permettent de déterminer la « politique » d'échantillonnage sur une période temporelle t en prenant en compte des données de la période précédente $t-1$. En effet, nous estimons que les données provenant des capteurs ont des comportements semblables pendant deux périodes successives de longueur fixée. Nous démontrons l'efficacité de nos algorithmes en les appliquant à des données de consommation électrique.
Cet article est organisé comme suit : la Section 2 présente un état de l'art sur l'échantillonnage dans les flux de données et les techniques d'alimentation des entrepôts de données. La Section 3 présente la formulation du problème, sa méthode de résolution est détaillée à la Section 4. Nous exposons les expérimentations effectuées pour valider l'approche à la Section 5. Enfin, nous concluons à la Section 6 en donnant nos perspectives de recherche.

2 Etat de l'art

2.1 Alimentation des entrepôts de données

Un entrepôt de données offre la possibilité de regrouper des données et de les exploiter (Jarke et al., 2003). Pour alimenter l'entrepôt de données, on utilise généralement un ETL (Extract, Transform and Load), outil décrivant les données, leur provenance et les transformations effectuées. Il permet d'agréger, de classer, de normaliser, et de nettoyer les données extraites. L'alimentation d'un entrepôt de données se fait de façon périodique suivant une périodicité définie par l'administrateur. Des décisions d'agrégation ou d'échantillonnage des

données sont faites à l'avance en fonction des besoins prévisibles d'analyse et de fouille de données, mais ces décisions peuvent s'avérer inadaptées si les besoins évoluent. Par ailleurs ces dernières années, un grand nombre d'applications sont apparues générant des quantités de données de façon distribuée. Cette apparition s'accompagne de difficultés liées essentiellement à la taille et à la distribution des données et des structures manipulées. Ceci pousse à revisiter les technologies d'entrepôts de données et d'analyse en ligne. En effet, l'architecture doit s'adapter à la distribution des données, à leur volumétrie croissante et à leur taux d'arrivée. L'acquisition, la modélisation, le nettoyage, l'interrogation et l'analyse doivent être étendus aux flux de données (data streams).
(Bauzer et al., 2006) aborde les problèmes rencontrés lors du traitement de données spatio-temporelles, mesurées en temps réel, où les sources de données correspondent à des capteurs qui sont placés sur les axes routiers d'une ville et transmettent à des centrales de données des mesures sur la circulation à chaque intervalle de temps. Un autre exemple de réseaux de capteurs est donné dans (Gonzalez et al., 2006) , il concerne les technologies RFID (Radio Frequancy IDentification) qui fournissent l'identification numérique des marchandises servant à les dépister. Dans un exemple cité dans cet article, on estime les données récoltées par RFID à 300 millions de tuples par jour sous la forme (Identifiant, date, position) si un détaillant possède 3000 magasins vendant 10000 produits par jour et par magasin (en moyenne un produit parcourt 10 emplacements avant sa vente). Ce grand volume de données ne pouvant pas être stocké entièrement, il est donc nécessaire d'effectuer un prétraitement à la volée avant leur enregistrement.
De nouvelles solutions technologiques capables de traiter de très grandes quantités d'information à la volée et en temps réel sont apparues ces dernières années. Plusieurs prototypes sont proposés dans la littérature (Abadi et al., 2005; Arasu et al., 2004; Zdonik et al., 2003). L'idée maîtresse de ces prototypes est de traiter le flux d'information alors qu'il est encore en circulation avant son enregistrement dans les entrepôts de données, ce qui permet d'accélérer les requêtes et les traitements de fouille qui seront ensuite appliqués à ces données.

2.2 Echantillonnage dans les flux de données

L'échantillonnage dans les flux de données s'appuie sur les techniques d'échantillonnage traditionnelles, mais requiert des innovations significatives pour parer au problème de la longueur infinie des flux. En effet, des techniques de fenêtrage sont utilisées pour s'adapter à la nature illimitée des données : une fenêtre définit un intervalle temporel exprimé soit en terme de durée (par exemple les 5 dernières minutes), soit sous forme logique exprimé en nombre de tuples (par exemple les 20 derniers éléments). Ces fenêtres peuvent être délimitées par des bornes fixes ou glissantes dans le temps.
Un algorithme classique en ligne proposé par Vitter en 1985 est largement utilisé dans le cadre des flux de données : l'échantillonnage réservoir (Vitter, 1985). Il permet d'obtenir un échantillon uniforme aléatoire avec une taille fixe et ne nécessite pas de connaître la taille du flux de données. L'échantillonnage réservoir est utile dans le cas d'insertion ou de mises à jour mais trouve ses limites à l'expiration des données dans une fenêtre glissante. En effet, dans ce type de fenêtre, les éléments ne faisant plus partie de la fenêtre courante deviennent invalides, et s'ils appartiennent à l'échantillon, il faut les remplacer. Des algorithmes permettant de tenir à jour un échantillon sur une fenêtre tout en préservant sa représentativité sont donc nécessaires. Plusieurs techniques ont été développées pour traiter le cas des fenêtres glissantes (logiques et

temporelles).
Une approche simple a été proposée dans (Babcock et al., 2002). Son principe est le suivant : on maintient un échantillon de type réservoir pour les premiers éléments du flux (première fenêtre) puis, quand un élément expire, on le remplace par le nouvel élément qui arrive. Cet algorithme maintient un échantillon aléatoire uniforme pour la première fenêtre et ne requiert pas beaucoup de mémoire, mais a l'inconvénient d'être hautement périodique. Pour remédier à ce défaut, une autre technique a été proposée : l'échantillonnage avec réserve. Son principe est le suivant : quand un élément arrive, on l'ajoute avec une probabilité donnée à un échantillon réserve ('backing sample') et on génère ensuite un échantillon aléatoire à partir de l'échantillon réserve. Quand un élément expire, on le supprime de la réserve.
Une autre méthode adaptée au flux de données est le tirage de Bernouilli (Brown et al., 2006). Cette méthode a le mérite d'être simple à mettre en oeuvre mais a un inconvénient majeur qui est la variabilité incontrôlable de la taille de l'échantillon. Plusieurs autres algorithmes ont été développés pour être appliqués aux fenêtres logiques (tel que l'échantillonnage en chaîne (Babcock et al., 2002), aux fenêtres temporelles (tel que l'échantillonnage par priorité (Babcock et al., 2002) ou pour des conditions particulières d'utilisation (tel que le réservoir avec fréquences exactes (Gibbons et al., 1998).
A notre connaissance, toutes ces techniques échantillonnent les flux de données individuellement. De plus, appliquées à des capteurs de données, ces techniques n'exploitent pas d'éventuelles possibilités de calcul au niveau de chaque capteur, ou encore de communication bidirectionnelle entre les capteurs et le système récoltant les données pour émettre et recevoir des ordres.

3 Formulation du problème

Soit n le nombre de capteurs reliés à un concentrateur et qui envoient une séquence de mesures temporelles en continu. Nous supposons dans le cadre de cet article que les flux de base sont réguliers (les données sont générées régulièrement). Nous découpons chaque flux en périodes (fenêtres temporelles) de même taille. Chaque fenêtre temporelle est constituée de p éléments (nous supposons constant le nombre d'éléments d'une fenêtre car le flux est régulier). Dans la suite de cet article, nous utiliserons le terme « courbe » pour qualifier chaque séquence temporelle. Soit m la borne inférieure du nombre de données que l'on souhaite prélever par courbe sur la fenêtre courante t, m signifie que nous récupérons au moins m éléments par courbe. Soit s le nombre d'éléments communicables au système central par les n capteurs sur la période t ($s < n * p$), c'est à dire le nombre de points que le système central peut accepter pendant la période.
Nous cherchons à déterminer un planning de collecte de données à effectuer pendant une période t (une journée par exemple) sur les n capteurs. Le problème consiste à trouver la meilleure « politique de construction de résumé » pour chaque capteur en respectant les contraintes du nombre maximal de données communicables au concentrateur et du nombre de données minimal à prélever par capteur.

Echantillonnage régulier

Nous échantillonnons les courbes à un pas j inférieur à $\lfloor \frac{p}{m} \rfloor$ ($\lfloor x \rfloor$ signifie partie entière de x). j correspond au 'saut' à effectuer entre deux points sélectionnés, par exemple $j = 2$ signifie que nous sélectionnons un point sur deux de la courbe. Cette méthode d'échantillonnage est qualifiée de « régulière » car les données prélevées sont équidistantes temporellement, nous effectuons un saut de j entre deux éléments échantillonnés.
Le problème posé consiste à trouver des pas d'échantillonnage pour chaque capteur en respectant les contraintes sur la limite du concentrateur et le nombre minimal de données à sélectionner par capteur. Les pas d'échantillonnage sont calculés à partir des points des courbes de la fenêtre temporelle $t-1$, et sont appliqués aux points de la période t.
Les pas d'échantillonnage doivent permettre de représenter la courbe initiale le plus finement possible. Le problème d'échantillonnage consiste donc à minimiser la somme des erreurs quadratiques (SSE pour Sum Square Error) entre la courbe d'origine C à la période $t-1$, et la courbe échantillonnée $\hat{C}$ en prenant en compte les contraintes citées ci-dessus. SSE se calcule de la façon suivante :
$SSE(C, \hat{C}) = \sum_{i=1}^{p} (c_i - \hat{c}_i)^2$, p est le nombre de points de la courbe C.

Echantillonnage irrégulier

Dans un objectif de compression, nous ne voulons conserver que les informations les plus pertinentes sur les courbes. Nous utilisons une technique d'échantillonnage irrégulier (à pas variable). En effet, nous envisageons de segmenter les courbes, il s'agit d'un moyen de découper une série chronologique en épisodes et d'associer une information à chacun de ces épisodes, le nombre d'épisodes étant fixé à l'avance. Les épisodes peuvent être de durées différentes d'où le terme « irrégulier ». La courbe segmentée est représentée par une fonction en « escalier ». Cette méthode est utile d'une part pour décrire et résumer l'information, et d'autre part car elle permet de réduire le bruit en lissant la courbe. La figure 2 donne un exemple d'une courbe de données de consommation électrique et 10 épisodes (en pointillés) générés par segmentation.
Soit une courbe C de p points, notés $c_1, c_2, ..., c_p$. Pour un découpage en k segments, on note

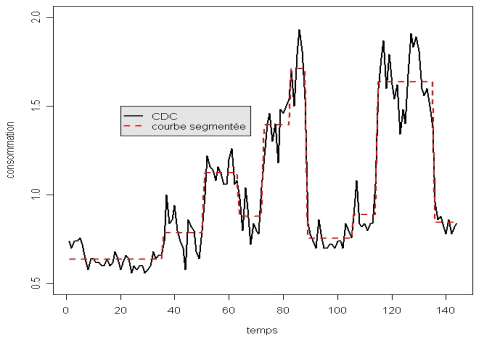

FIG. 2 – *Une courbe et sa segmentation en 10 épisodes*

$c_{1l}, ..., c_{il}, ..., c_{n_l l}$ la séquence appartenant au segment l avec n_l le nombre de points du segment l. Pour tout c_{il}, on note $\hat{c}_l$ le représentant du segment l. La courbe segmentée est alors représentée par $\hat{C} = ((\hat{c}_1, n_1), (\hat{c}_2, n_2), ..., (\hat{c}_k, n_k))$ où chaque épisode l est défini par $\hat{c}_l$ qui est le représentant numérique des n_l points de la courbe initiale C.
La valeur numérique $\hat{c}_l$ associée à un segment doit représenter l'ensemble des valeurs des points pris au cours du segment en question. La segmentation d'une courbe est donc la recherche de l'ensemble des segments et des valeurs numériques associées à ceux-ci de façon à optimiser un critère donné. Généralement, le critère utilisé est la somme des erreurs quadratiques calculée par l'expression suivante : $\sum_{l=1}^{k} \sum_{i=1}^{n_l} (c_{il} - \hat{c}_l)$. Dans ce cas, le représentant numérique est la moyenne des points de la courbe initiale qui définissent un segment.
(Hugueney, 2003) énumère plusieurs algorithmes de recherche de segmentation. Nous avons utilisé dans le cadre de cet article la stratégie optimale employant la programmation dynamique (Bellman, 1961). En effet, l'erreur de modélisation globale étant la somme d'erreurs quadratiques indépendantes, la segmentation optimale en k segments est constituée d'un premier segment suivi de la segmentation optimale en $k-1$ segments à partir de la fin du premier segment. La méthode utilisée est détaillée dans (Hugueney, 2003).
Le problème posé ici consiste à déterminer le nombre de segments pour chaque compteur en respectant -comme pour le cas de l'échantillonnage régulier- le nombre de données communicables au concentrateur pendant une période donnée ainsi que le nombre minimal de données à prélever par capteur. Nous utilisons les courbes de la période $t-1$ pour déterminer la politique de segmentation à appliquer à la période t. La segmentation doit permettre de représenter la courbe initiale le plus précisément possible.

Formulation du problème

Le problème d'échantillonnage consiste à trouver une approximation $\hat{C}_i$ de C_i (courbe provenant du capteur i), telle que :

$$Arg \min_{\hat{C}_i} (\sum_{i=1}^{n} SSE(C_i, \hat{C}_i))$$

où n est le nombre de capteurs et SSE est la somme des erreurs quadratiques entre C_i et le modèle $\hat{C}_i$.
Cette minimisation des SSE doit se faire avec les contraintes :

- $\sum_{i=1}^{n} \left\lfloor \frac{p}{j_i} \right\rfloor \leq s$, s est la limite des données à communiquer.
- $\forall i \in 1, 2, ..., n$, $1 \leq j_i \leq \left\lfloor \frac{p}{m} \right\rfloor$.

j_i correspond au pas d'échantillonnage dans le cas de l'échantillonnage régulier et au nombre de segments à appliquer à la courbe d'indice i dans le cas de l'échantillonnage irrégulier.
Une fois les valeurs j_i déterminées, se pose le problème d'envoi de données au premier pas de temps. En effet, tel que le problème est formulé, tous les capteurs envoient simultanément une valeur au premier pas de temps, et nous dépassons ainsi la limite de la bande passante. Nous ne nous intéressons pas ici à ce problème qui peut facilement être contourné en répartissant automatiquement les échanges dans le temps.

4 Méthode de résolution

La méthode la plus immédiate pour déterminer les pas d'échantillonnage (ou le nombre de segments) est de partager également la bande passante sur les différentes courbes, les pas d'échantillonnage (ou le nombre de segments) étant les mêmes pour toutes les courbes. Cette solution respecte bien les contraintes posées mais ne minimise pas la somme des erreurs quadratiques entre les courbes initiales et les courbes échantillonnées (segmentées). De plus, les courbes présentant des fluctuations différentes, les pas d'échantillonnages « fixes » pourraient sur-échantillonner une courbe ou au contraire la sous-échantillonner. Pour résoudre le problème de minimisation des SSE, nous l'avons modélisé sous forme d'un problème d'optimisation linéaire. Nous disposons de *n* courbes, chacune contenant *p* points. Voici comment nous avons procédé pour modéliser le problème.

Echantillonnage régulier

Chaque courbe doit être échantillonnée à un pas inférieur ou égal à $\lfloor \frac{p}{m} \rfloor$ (m étant inférieur à p). Nous avons donc la possibilité de choisir un pas entier allant de 1 (nous récupérons tous les points de la courbe) à $\lfloor \frac{p}{m} \rfloor$ (les points échantillonnés sont distancés par $\lfloor \frac{p}{m} \rfloor$). Notons $m' = \lfloor \frac{p}{m} \rfloor$, nous calculons une matrice $W_{n*m'}$ de n lignes et m' colonnes. Un élément w_{ij} de la matrice correspond à la somme des erreurs quadratiques obtenue si on applique un pas d'échantillonnage *j* à la courbe d'indice *i*.

Echantillonnage irrégulier

Le nombre de segments minimal que nous pouvons appliquer à une courbe est m et p est le nombre maximal des segments (tous les points de la courbe sont récupérés). Nous pouvons affecter une valeur entière j à une courbe allant de 1 (nous récupérons tous les points de la courbe) à m (nous découpons la courbe en m segments). Nous calculons de la même façon que précédemment une matrice W_{n*m} de n lignes et m colonnes. Un élément w_{ij} de la matrice correspond à la somme des erreurs quadratiques obtenue si on découpe de façon optimale la courbe d'indice i en $\left\lfloor \frac{p}{j} \right\rfloor$ segments.

Généralisation

Le problème à résoudre peut s'énoncer comme suit :

$$\text{Minimiser } \sum_{i=1}^{n} \sum_{j=1}^{m'} (W_{ij} \times X_{ij})$$

sous les contraintes :

$$\begin{cases} X_{ij} = 0 \text{ ou } 1 & \\ \sum_{j=1}^{m'} X_{ij} = 1 & \text{i de 1 à n} \\ \sum_{i=1}^{n} \sum_{j=1}^{m'} (\left\lfloor \frac{p}{j} \right\rfloor \times X_{ij}) \leq s & \text{i de 1 à n} \end{cases}$$

Il s'agit d'un problème d'affectation de pas d'échantillonnage (ou de nombre de segments) sur les différentes courbes en respectant les contraintes ci-dessus. Une variable X_{ij} à 1 signifie que nous affectons le pas d'échantillonnage j (ou $\left\lfloor \frac{p}{j} \right\rfloor$ segments) à la courbe d'indice i . La deuxième contrainte $\sum_{j=1}^{m'} X_{ij} = 1$ impose une seule valeur de j par courbe (un seul pas d'échantillonnage ou un seul nombre de segments par courbe). Enfin, la troisième contrainte signifie que le nombre de données à communiquer au système central ne doit pas dépasser le seuil imposé s.
Pour résoudre ce problème, nous utilisons la méthode du simplexe appliquée aux problèmes linéaires à variables réelles. Le simplexe est couplé avec la méthode Branch And Bound (Séparation-Evaluation en français) afin d'atteindre des variables entières. Le programme LP_Solve permet de résoudre des problèmes d'optimisation linéaires à variables réelles et/ou entières . Le lecteur peut se référer à (Gondran et al., 1979; lpsolve) pour des informations sur ces méthodes.

5 Expérimentations

Avec le déploiement de compteurs communicants chez les clients des fournisseurs d'électricité, les consommations d'énergie électrique pourront être télérelevées à des pas de temps pouvant aller jusqu'à la seconde. Ceci permettra d'effectuer des opérations tels que la facturation, l'agrégation, le contrôle,... Les courbes de consommation électrique (aussi appelées courbes de charge) sont aussi utilisées pour étudier la consommation en électricité d'un client dans le temps. Il s'agit de l'évolution de la consommation d'énergie entre deux instants au cours du temps. Nos expérimentations ont été faites sur un jeu de données de courbes de charge relevées à pas de 10 minutes pendant une journée (144 relèves par compteur électrique). Il s'agit d'un fichier de 1000 compteurs électriques. Les courbes de charge ont été normalisées.

5.1 Méthode d'interpolation

Echantillonnage régulier :

La somme des erreurs quadratiques dépend de la façon dont nous construisons la courbe échantillonnée (dans le cas de l'échantillonnage régulier). Nous avons choisi deux méthodes et les avons comparées entre elles. Les résultats sont exposées à la section 5.2.
Soient une courbe *C* contenant *p* points $C = \{c_1, c_2, ..., c_p\}$ et $\hat{C}$ sa courbe échantillonnée à un pas j, $\hat{C} = \{\hat{c}_1, \hat{c}_2, ..., \hat{c}_p\}$.
La première méthode construit la courbe échantillonnée sous forme de fonction en escalier, les points compris entre deux données sélectionnées successives c_a et c_b prennent la valeur de la première donnée c_a. C'est à dire que la relation entre C et $\hat{C}$ est donnée par :

$$\hat{c}_i = \begin{cases} c_i & \text{si } i \; modulo \; j = 1 \\ c_{i-1} & \text{sinon} \end{cases}$$

La deuxième méthode construit la courbe échantillonnée par interpolation linéaire, les valeurs des points compris entre deux données sélectionnées successives c_a et c_b sont calculées

par interpolation linéaire en utilisant les valeurs de c_a et c_b :

$$\hat{c}_i = \begin{cases} c_i & \text{si } i \; modulo \; j = 1 \\ f(i) & \text{sinon} \end{cases}$$

Soient c_a et c_b les deux points échantillonnés de C tel que $a < i < b$. La fonction f est la droite d'interpolation linéaire entre les deux points c_a et c_b, et a pour équation :
$f(i) = \frac{i-b}{a-b} c_a - \frac{i-a}{a-b} c_b$

Echantillonnage irrégulier :

Pour reconstruire la courbe segmentée (en escalier) à partir des moyennes de chacun des épisodes, nous avons besoin aussi du nombre de points de chaque épisode. A titre d'exemple, si la courbe initiale C a été segmentée en k segments, le concentrateur a besoin des données $(\hat{c}_1, n_1), (\hat{c}_2, n_2), ..., (\hat{c}_k, n_k)$, $\hat{c}_l$ étant la moyenne de l'épisode l constituée de n_l points. Le compteur doit donc envoyer $2 * j$ données au concentrateur, ce qui sera pris en compte dans la vérification de la contrainte de seuil s.

5.2 Résultats

Nous avons considéré $m = 7$ ($m' = \lfloor \frac{144}{7} \rfloor = 20$) ce qui signifie qu'une courbe échantillonnée sera constituée d'au moins 7 valeurs si elle est construite de façon régulière et de 3 valeurs si elle est construite de façon irrégulière. En effet, quand on segmente une courbe, on envoie les moyennes des épisodes accompagnées du nombre de points de la courbe initiale qui constituaient l'épisode. Nous rappelons qu'une courbe de charge dans notre exemple est constituée de 144 points. Nous avons testé plusieurs valeurs de bande passante (seuil s) et nous avons dressé un tableau récapitulatif des résultats obtenus (tableau 1). Chaque colonne du tableau correspond à une valeur du seuil s. Les méthodes de résumé qui ont été testées sont les suivantes :

- Echantillonnage régulier : 'Interpolation linéaire', 'escalier' et 'fixe'(les courbes sont toutes échantillonnées au même pas j) ;
- Echantillonnage irrégulier : 'Segmentation variable' obtenue par optimisation et 'segmentation fixe' en utilisant le même nombre de segments pour toutes les courbes de charge.

Nous avons calculé la somme des erreurs quadratiques pour chacune des méthodes, la moyenne des j affectés par optimisation (pas d'échantillonnage pour les méthodes régulières et le nombre de segments pour les méthodes irrégulières), l'écart-type pour mesurer la dispersion des j autour de la moyenne, et enfin la moyenne des différences des pas d'échantillonnage entre la méthode 'escalier' et la méthode 'interpolation linéaire'.

- Echantillonnage régulier : Les résultats du tableau 1 font apparaître que l'optimisation permet de diminuer considérablement la somme des erreurs quadratiques que ce soit avec la méthode en 'escalier' ou avec l'interpolation linéaire par rapport à l'échantillonnage au pas 'fixe'. Par exemple, en utilisant un seuil de 72000, l'optimisation avec 'Interpolation Linéaire' nous permet de minimiser l'erreur quadratique de 99% par rapport à l'échantillonnage 'fixe'.

Quoique les moyennes des pas d'échantillonnage par optimisation se rapprochent des

seuil	144000	72000	48000	28800	20571	14400	9600
SSE opt esc	0	33.26	103.11	321.8	662.42	1511.1	4470.3
SSE opt IL	0	13.88	47.88	154.51	330.83	793.29	2336.9
SSE opt seg	0	13.29	42.86	132.16	271.01	607.5	1730.6
SSE fixe esc	0	2770.8	4329	8040	9313	13157	16048
SSE fixe IL	0	1956	2821	4245	5501	7024	9087
SSE fixe seg	0	505.8	860.8	1535	2042.6	2761.5	4677.1
moyenne pas	1	2	3	5	7	10	15
moyenne segs	144	36	24	14	10	7	3
ecart-type pas esc	0	3.99	5.48	7.23	8.24	8.62	7.19
ecart-type pas IL	0	4.11	5.63	7.46	8.57	9.18	7.93
ecart-type segs	0	3.82	5.18	6.87	7.92	8.32	7.12
moyenne pas esc-IL	0	0.56	1.3	1.67	1.52	2.3	2.5

TAB. 1 – *tableau récapitulatif. SSE : Somme de l'écart quadratique. opt : pas d'échantillonnage (nb segments) obtenus par optimisation, et fixe : même pas d'échantillonnage (nb segments) pour les CDCs. esc : courbe échantillonnée en escalier et IL : par Interpolation Linéaire. seg : segmentation La dernière ligne correspond à la moyenne des différences entre les pas d'échantillonnage en escalier et avec IL.*

pas 'fixes', nous observons des valeurs élevées des écarts-types. Les pas d'échantillonnage sont largement écartés de la moyenne. La figure 3 est un exemple de distribution des pas d'échantillonnage ('escalier' et 'interpolation linéaire') pour un seuil de 14400. Notons que les pas d'échantillonnage par optimisation sont répartis sur les différentes valeurs des pas avec une concentration au niveau des limites (*j=1* et *j=20*), alors qu'en échantillonnage 'fixe' le pas d'échantillonnage est *j=10*. Ce fait permet de confirmer l'importance de sélectionner les pas d'échantillonnage par optimisation.

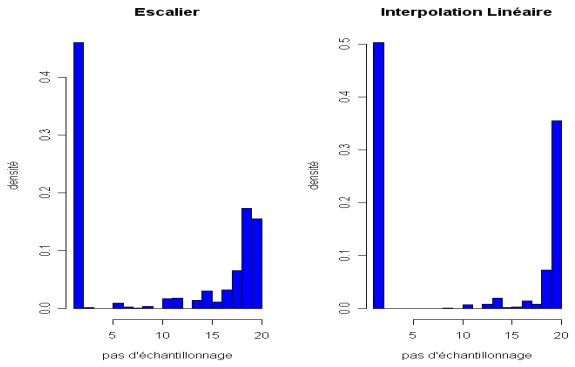

FIG. 3 – *Distribution des pas d'échantillonnage*

Il est aussi intéressant de remarquer qu'il n'y a pas de grandes différences en moyenne entre les pas d'échantillonnage affectés par la méthode escalier et la méthode utilisant l'interpolation linéaire. Nous pouvons donc utiliser indifféremment l'une ou l'autre dans nos prochaines expérimentations.

– Echantillonnage irrégulier : Le nombre de segments a été choisi parmi l'ensemble $\left\{\left\lfloor\frac{p}{1}\right\rfloor, \left\lfloor\frac{p}{2\times 2}\right\rfloor, \left\lfloor\frac{p}{2\times 3}\right\rfloor \ldots, \left\lfloor\frac{p}{2\times m'}\right\rfloor\right\}$, $p = 144$ et $m' = 20$. Au vu des résultats du tableau 1, nous remarquons que la segmentation permet de mieux minimiser la somme des erreurs quadratiques par rapport à l'échantillonnage régulier que ce soit avec la méthode escalier ou en utilisant l'interpolation linéaire. La segmentation donne des résultats meilleurs malgré le nombre inférieur de données récupérées par période de temps. La différence est plus apparente pour des valeurs de seuil élevées auquel cas il est préférable de segmenter les courbes que de les échantillonner régulièrement. A titre d'exemple, pour un seuil de 9600, la segmentation a minimisé la somme des erreurs quadratiques de 57% par rapport à l'échantillonnage escalier et de 26% par rapport à l'échantillonnage avec interpolation linéaire . Nous remarquons que les écart-types sont aussi élevés dans le cas de la segmentation que dans le cas de l'échantillonnage régulier. Les nombres de segments affectés aux courbes de charges sont bien dispersés autour de la moyenne.

5.3 Traitement global du flux

Les expérimentations précédentes évaluent l'erreur commise sur une période t à partir d'une optimisation effectuée sur cette même période. L'approche globale proposée consiste à appliquer sur une période t le résultat d'une optimisation réalisée sur la période $t-1$. Nous présentons ici la méthode retenue pour le traitement en continu des flux ainsi que de premières expérimentations réalisées sur 140 courbes de charge disponibles sur une période d'une année.

A la période $t = 0$, le concentrateur n'a aucune connaissance des données des compteurs qui lui sont attachés. Celui-ci demande aux compteurs d'envoyer $\left\lfloor\frac{s}{n}\right\rfloor$ valeurs chacun tout au long de la première période, c'est un échantillonnage « fixe ». A l'expiration de la période, les compteurs calculent les erreurs correspondant aux deux méthodes d'échantillonnage (régulier et irrégulier) et envoient le minimum des erreurs entre ces deux méthodes au concentrateur pour constituer la matrice des erreurs $W_{n*m'}$. Le concentrateur applique le programme d'optimisation pour trouver les j correspondant aux pas d'échantillonnage dans le cas régulier et au double du nombre de segments dans le cas irrégulier. Ensuite, il envoie le résultat de l'optimisation aux compteurs. Si le SSE en appliquant un pas d'échantillonnage j au compteur est inférieure au SSE en segmentant la courbe en $\left\lfloor\frac{p}{j\times 2}\right\rfloor$, alors le compteur décide d'appliquer un échantillonnage régulier et envoie les données de consommations électriques au concentrateur. Dans le cas inverse, le compteur envoie les moyennes des segments accompagnés du nombre de points constituant le segment. A la fin de la deuxième période $t = 1$, les compteurs envoient de la même façon que précédemment les SSEs pour mettre à jour la matrice des erreurs. Ce processus continue tant que les flux de données arrivent en ligne. Le schéma 4 résume le processus d'échange entre le concentrateur et les capteurs.

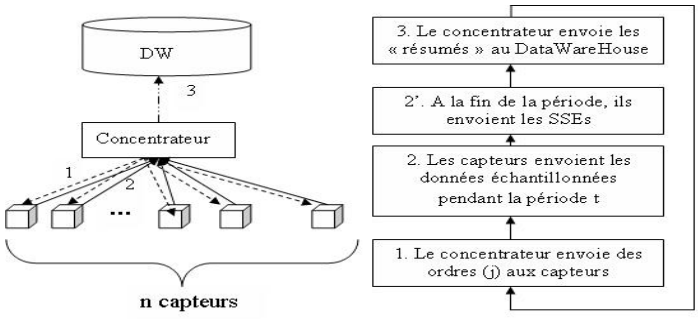

FIG. 4 – *Processus d'échange entre le concentrateur et les capteurs*

Les expérimentations ont porté sur jeu de données composé de 140 courbes de consommation électrique relevées à un pas de temps de 30 minutes sur une année (365 jours). Le tableau 2 récapitule les résultats obtenus.

La durée utilisée pour la période t est d'une journée. La phase d'optimisation utilise les données de la journée précédente et les pas d'échantillonnage sont appliqués à la journée courante, ceci tout au long de l'année. Nous calculons les moyennes des erreurs d'échantillonnage par interpolation linéaire et par fonction en escalier obtenues sur toute l'année, et les comparons aux moyennes des erreurs obtenues par échantillonnage 'fixe'. Les résultats montrent que la méthode est très performante. En effet, l'optimisation de l'échantillonnage sur le flux permet de diminuer d'au moins 10% par rapport à la méthode 'fixe' les erreurs d'échantillonnage de plus de 90% des jours de l'année, pour toutes les valeurs de seuil expérimentées. La figure 5 montre l'évolution des SSEs durant un mois (30 jours) pour un seuil de 1344. Une périodicité des erreurs apparaît sur la courbe correspondant à l'évolution des SSEs en appliquant les fréquences d'échantillonnage obtenues par optimisation pendant la journée précédente (courbe opt-1). Ceci est dû au cycle hebdomadaire de la consommation d'électricité (5 jours ouvrables avec une consommation globalement stable et le week end où la consommation change de niveau). Dans la méthode, nous déterminons les fréquences d'échantillonnage pour les journées du lundi en se basant sur les données du dimanche et celles du samedi par les données du vendredi. Nous avons expérimenté le cas où la phase d'échantillonnage sur les données de la journée courante j_c se fait grâce aux fréquences d'échantillonnage calculées à partir de la courbe de la journée $j_c - 7$. Au vu de la courbe opt-7 de la figure 5 nous remarquons que cette méthode permet de 'lisser' la courbe de l'évolution des erreurs d'échantillonnage et ainsi diminuer les erreurs dûes à la périodicité hebdomadaire de la consommation électrique.

Nous avons également expérimenté une durée de la période t égale à la semaine, en suivant l'évolution des SSEs durant les 52 semaines de l'année. Dans ce cas aussi, l'approche permet de diminuer les erreurs d'échantillonnage par rapport à l'échantillonnage fixe, et ceci pour plus de 90% des semaines de l'année dans le cas de l'interpolation linéaire et de 96% dans le cas de la méthode en escalier.

seuil	6720	3360	2240	1344	960	672	448
moyenne pas	1	2	3	5	7	10	15
moy SSE opt-1 IL	0	22	45	104	192	371	814
moy SSE fixe IL	0	148	275	485	830	1151	1719
%j $<$ 10% SSE IL	0	94%	95.8%	95%	96.4%	95%	95.6%
moy SSE opt-1 esc	0	41	75	172	312	551	1224
moy SSE fixe esc	0	279	554	1156	1818	2704	3465
%j $<$ 10% SSE esc	0	96%	99%	100%	100%	100%	100%
moy SSE opt-7 IL	0	16	36	79	143	285	716
moy SSE fixe IL	0	149	275	485	830	1151	1722
%j $<$ 10% SSE IL	0	98%	99%	99.7%	99.4%	99.4%	99.7%
moy SSE opt-7 esc	0	29	62	134	226	421	1049
moy SSE fixe esc	0	279	554	1158	1821	2709	3475
%j $<$ 10% SSE esc	0	99%	99%	100%	100%	100%	100%
moy SSE opt-1 IL	0	190	352	926	1780	3149	6637
moy SSE fixe IL	0	1127	2119	3995	6793	9438	13725
%sem $<$ 10% SSE IL	0	94%	96%	94%	90%	94%	94%
moy SSE opt-1 esc	0	353	621	1585	2911	4688	10364
moy SSE fixe esc	0	2130	4389	9165	14798	22009	26882
%sem $<$ 10% SSE esc	0	96%	100%	100%	100%	100%	100%

TAB. 2 – *tableau récapitulatif. opt-i : pas d'échantillonnage obtenus par optimisation à la période t-i. moy : la moyenne des SSEs obtenus pendant toute l'année jour par jour ou semaine par semaine ($*10^3$). %j $<$ 10% SSE (%sem $<$ 10% SSE) correspond au pourcentage de jours (semaines) dont le SSE opt-1 (opt-7) est inférieur de 10% au SSE obtenu par échantillonnage fixe.*

6 Conclusion et perspectives

La mise en oeuvre de l'échantillonnage appliqué à des courbes a permis de montrer que l'affectation des pas d'échantillonnage par optimisation linéaire permet de réduire significativement les erreurs d'échantillonnage par rapport à un échantillonnage à pas 'fixe'. Notre problématique s'insère dans le cadre d'échantillonnage des flux de données en spécifiant des fenêtres temporelles glissantes (courbe d'une période t).
Nos expérimentations en cours s'effectuent sur un jeu de données plus grand (1000 courbes de charge correspondant à une période d'un an), nous testons plusieurs tailles de fenêtres temporelles (journée, semaine, ...) et nous suivons l'évolution des erreurs dans le temps en utilisant les deux méthodes d'échantillonnage (régulier et irrégulier).
Un autre axe de recherche est d'imposer un nombre minimum de courbes échantillonnées par un pas d'échantillonnage donné (nombre de segments donné). Ceci nous permettra d'effectuer des calculs et des estimations plus précis sur les corrélations entre les courbes. Nous envisageons aussi d'appliquer d'autres mesures d'erreur d'échantillonnage telles que les normes L_1 et L_∞ pour éviter les erreurs ponctuelles qui peuvent être importantes pour certaines courbes.

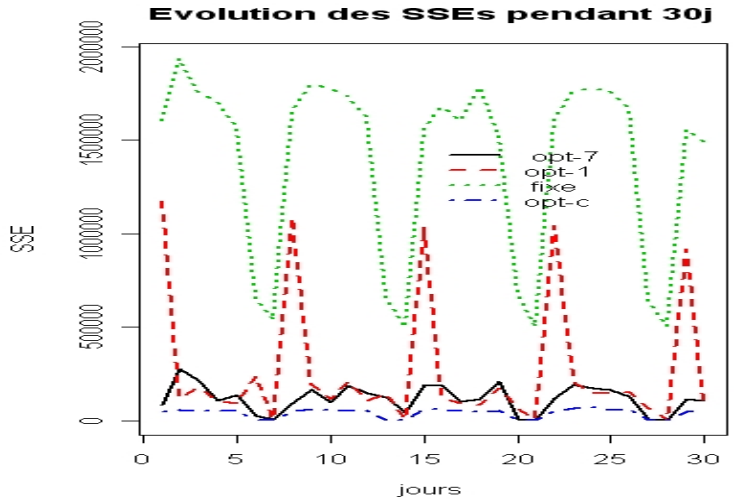

FIG. 5 – *Evolution des SSEs pendant 30jours. opt-7 : Optimisation au jour j et echantillonnage à j+7. opt-1 : Optimisation au jour j et echantillonnage à j+1. opt-c : Optimisation et echantillonnage à j (jour courant).*

On peut aussi prendre en compte les erreurs inter-courbes en minimisant le maximum d'erreur des courbes. Ces problèmes d'optimisation sont en cours de résolution.

Références

Abadi D.J., Ahmad Y., Balazinska M., Cetintemel U., Cherniack M., Hwang J.H., Lindner W., Maskey A.S., Rasin A., Ryvkina E., Tatbul N., Xing Y., Zdonik S, (2005). The Design of the Borealis Stream Processing Engine, 2nd Biennial Conference on Innovative Data Systems Research (CIDR'05), pp.277-289, Asilomar, CA, January 2005.

Arasu A., Babcock B., Babu S., Cieslewicz J., Datar M., Ito K., Motwani R., Srivastava U., Widom J., (2004). STREAM : The Stanford Data Stream Management System, available at http ://dbpubs.stanford.edu/pub/2004-20, 2004.

Babcock B., Datar M., Motwani R., (2002). Sampling From a Moving Window Over Streaming Data, 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002), pp.633-634, 2002.

Babcock B., Babu S., Datar M., Motwani R., Widom J., (2002). Models and issues in data stream systems. In Symposium on Principles of Database Systems, pp.1-16. ACM SIGACT-SIGMOD, 2002.

Bauzer-Medeiros C., Carles O., Devuyst F., Hébrail G., Hugueney B., Joliveau M., Jomier G., Manouvrier M., Naïja Y., Scemama G., Steffan L., (2006). Vers un entrepôt de données pour le trafic routier, 2ème Journée Francophone sur les Entrepôts de Données et l'Analyse en

ligne (EDA 2006), Versailles (France), pp.119-137, Juin 2006, Revue des Nouvelles Technologies de l'Information - RNTI-B2.

Bellman R., (1961). On the approximation of curves by line segments using dynamic programming, Communications of the ACM, VOL.4, N.6, p.284, Juin 1961.

Brown P.G., Haas P.J., (2006). Techniques for warehousing of sample data, Proceedings of 22nd IEEE International Conference on Data Engineering, 2006.

Gibbons P.B., Matias Y., (1998). New sampling-based summary statistics for improving approximate query answers, Proceedings of ACM SIGMOD, pp.331-342, Seattle, WA, USA, June 1998.

Golab L., Özsu M.T., (2003). Data stream management issues - a survey, Technical report CS 2003-08, University of Waterloo, Waterloo, Canada, April 2003.

Gondran M., Minoux M., (1979). Graphes et algorithmes. Eyrolles, Paris 1979.

Gonzalez H., Han J., Li X., Klabjan D., (2006). Warehousing and Analyzing Massive RFID Data Sets, Proceedings of 22nd IEEE International Conference on Data Engineering (ICDE), 2006.

Hugueney B., (2003). Représentation symbolique de courbes numériques, Thèse de doctorat de l'Université Paris 6, 2003.

Jarke M., Lenzerini M., Vassiliou Y., Vassiliadis P., (2003). Fundamentals of data warehouses, 2nd edition, Springer, 2003.

http ://lpsolve.sourceforge.net/

Muthukrishnan S., (2005). Data streams : algorithms and applications, In Foundations and Trends in Theoretical Computer Science, Volume 1, Issue 2, August 2005.

Vitter J.S., (1985). Random sampling with a reservoir, ACM Trans. Math. Softw. 11, 1, pp.35-57, 1985.

Zdonik S., Stonebraker M., Cherniack M., Cetintemel U., Balazinska M., Balakrishnan H., (2003). The Aurora and Medusa Projects, Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, pp.3-10, March 2003.

Summary

Data warehouses are more and more supplied with data produced by a large number of distributed sensors in many applications: medicine, army, road traffic, weather forecast, utilities like electric power suppliers. Such data are largely distributed and produced continuously as data streams. We propose an approach to optimize the sampling rate to be applied to each data source, to be used for the load of such data in data warehouses in a data stream framework. Experiments done on real electric power consumption data are reported and show the efficiency of the proposed approach.