Résumé généraliste de flux de données

Collectif d'Auteurs Midas, Axis Inria -

Résumé généraliste de flux de données

Projet ANR MIDAS ANR07-MDCO-008

Collectif d'auteurs participant au projet MIDAS*

*contacts:
Georges Hébrail ou Christine Potier,
Dept INFRES, TélécomParis Tech
46 rue Barrault
75634 Paris Cedex 13
georges.hebrail@telecom-paristech.fr
ou
christine.potier@telecom-Paristech.fr
http://midas.enst.fr

Résumé. Lorsque le volume des données est trop important pour qu'elles soient stockées dans une base de données, ou lorsque leur taux d'arrivée est rapide, les Systèmes de Gestion de Flux de Données (SGFD) permettent de capturer des flux d'enregistrements structurés et de les interroger à la volée par des requêtes permanentes (exécutées de façon continue). Mais les SGFD ne conservent pas l'historique des flux qui est perdu à jamais. Cette communication propose une définition formelle de ce que devrait être un résumé généraliste de flux de données. La notion de résumé généraliste est liée à la capacité de répondre à des requêtes variées et de réaliser des tâches variées de fouille de données, en utilisant le résumé à la place du flux d'origine. Une revue de plusieurs approches de résumés est ensuite réalisée dans le cadre de cette définition. Enfin, l'intégration de telles approches de résumés dans un SGFD est discutée, à la fois en termes de construction et d'utilisation.

1 Introduction

Les nombreux travaux récents relatifs aux flux de données ont permis de dégager clairement les grandes différences entre "traitement des flux de données" et "traitement des bases de données". Une des différences structurantes est que le traitement des flux de données repose sur l'exécution de requêtes continues (valables sur la durée du flux) sur des données volatiles (potentiellement infinies par nature, le flux ne peut pas être stocké) alors que le traitement des bases de données repose sur l'exécution de requêtes volatiles ("one shot") sur des données persistantes (Golab et Özsu 2003).

A l'évidence, une requête continue sur un flux de données ne peut fournir de réponse que sur la fenêtre temporelle nécessairement limitée qui a été posée sur le flux. Il est dans la nature des données volatiles de n'être plus disponibles pour les analyses après leur expiration, ce qui suppose de poser a priori sur un flux l'ensemble des requêtes dont on pourra avoir besoin par la suite. Il est clairement hors de question de poser des requêtes a posteriori sur le flux. Ici "a posteriori" doit être entendu comme portant sur des données ayant expiré.

Pour pallier cet inconvénient, différents algorithmes et structures de données ont été proposés dans la littérature, souvent (mais pas toujours) sous le nom de "résumé" ("summary")

sans toutefois qu'émerge une notion cohérente de ce qu'est ou devrait être le résumé d'un flux de données.

Dans la suite de cet article, nous utiliserons le terme de "résumé d'un flux de données" aussi bien pour l'algorithme que pour la structure de données résultant de l'algorithme.

L'objectif de ce travail est d'introduire la notion de "résumé généraliste" dont l'objectif est de permettre l'exécution a posteriori de requêtes sur le flux, de définir cette notion précisément et d'analyser comment s'y rapportent certaines techniques de "résumé" de flux de la littérature. On conclura par une perspective sur l'apport et l'usage des résumés dans un système de gestion de flux de données.

2 Définition d'un résumé généraliste de flux de données

2.1 Définition d'un flux de données

Un flux de données, noté F, est une suite infinie d'événements ordonnés dans le temps, chaque événement étant caractérisé par :

- l'identifiant de l'objet ayant émis l'événement (typiquement une adresse dans un espace A non nécessairement connu a priori);
- l'estampille temporelle de l'événement (temps physique ou temps logique, cela dépendra du flux). Soit T l'espace des estampilles temporelles;
- la description de l'événement, qui peut comporter des descripteurs quantitatifs ou qualitatifs ; formellement, l'espace de description est un produit cartésien $D = R^p \times Q$ où R^p est l'espace de description des variables quantitatives et Q est l'espace des variables qualitatives qui pourrait simplement être décrit par un multiset de valeurs sur un alphabet (non nécessairement connu a priori).

A l'exception de l'estampille temporelle, les différents champs peuvent être non renseignés.

2.2 Notion idéale d'un résumé généraliste

Idéalement, un résumé généraliste est une structure de données mise à jour au fur et à mesure de l'arrivée des éléments du flux, et permettant de répondre a posteriori et approximativement à n'importe quelle requête portant sur le flux, de façon optimale compte tenu de contraintes de ressources. Cette structure doit permettre également de calculer des bornes sur la précision des réponses approchées à ces requêtes.

Cette définition idéale appelle quelques commentaires. Sur les contraintes de ressources tout d'abord : ces contraintes sont au moins de trois types, contraintes sur la mémoire disponible pour stocker le résumé, contrainte sur le temps de construction / mise à jour du résumé et contrainte sur le temps d'exécution d'une requête sur le résumé.

On se contente ici d'envisager un résumé sur un flux unique ; les notions de contrainte de communication (bande passante) dont il faut tenir compte dans le cas du traitement distribué de flux de données seront donc ignorées.

L'existence de ces contraintes et le caractère potentiellement infini du flux de données rendent inévitable une dégradation de l'information présente dans le résumé ; on ne peut qu'imposer que cette dégradation se fasse de façon "optimale".

On remarque que le résumé devant être à même de traiter toutes les requêtes, la notion d'optimalité est à définir sans faire référence à des requêtes particulières ; la possibilité de calculer des bornes sur les approximations obtenues des réponses aux requêtes en fonction des contraintes est une exigence supplémentaire, portant sur le résumé et qui permet de l'utiliser de façon contrôlée.

On note finalement que cette définition idéale impose de répondre à toutes les requêtes, ce qui distingue d'emblée la notion de "résumé généraliste" de la notion de "sketch" ou "synopsis" qui sont des structures de données conçues pour répondre (de façon optimale sous contrainte de ressources) à un type particulier de requête (requête COUNT pour le Count Min Sketch (Cormode et Muthukrishnan 2004), requête COUNT DISTINCT pour le sketch de Flajolet-Martin (Flajolet et Martin 1985), par exemple).

2.3 Définition réaliste d'un résumé généraliste

2.3.1 Définition

Une définition "réaliste" vise en particulier à guider l'inévitable perte d'information entre le résumé et le flux. Une première proposition est de restreindre la classe des requêtes concernées par le résumé aux seules requêtes de type SELECT AND COUNT où la sélection (prise ici au sens de *filtrage*) peut se faire sur toutes les variables, à l'exclusion des identifiants; les requêtes ponctuelles portant sur tel ou tel objet sont donc exclues du champ du résumé généraliste.

Cette restriction laisse une très grande expressivité au résumé généraliste : en effet, la capacité à répondre exactement à toute requête de type SELECT AND COUNT sur le temps et les variables descriptives revient à connaître exactement la densité jointe sur l'espace $T \times D$. En d'autres termes, le résumé généraliste permet une estimation approchée de la densité jointe. Alternativement, une estimation approchée de la densité jointe permet de répondre approximativement à toute requête de type SELECT AND COUNT.

Un résumé généraliste est donc une structure de données permettant une approximation optimale de la densité jointe sur $T \times D$, sous les contraintes d'espace mémoire, de temps de mise à jour / construction et d'exécution des requêtes.

La notion d'optimalité sous contrainte d'espace-mémoire peut être abordée en théorie sous l'angle de la théorie du Minimum Description Length (MDL) (Gründwald 2007). En dépit d'inconvénients pratiques évidents (l'optimum MDL est non calculable, la qualité de l'approximation ne décroît pas de façon monotone quand la longueur de description diminue, voir Adriaans et Vitanyi 2007), l'approche MDL reste une heuristique séduisante pour définir l'optimalité sous contrainte d'espace-mémoire.

Une solution "pure MDL" permet en particulier de mettre en évidence les interactions complexes entre les contraintes d'espace-mémoire et les contraintes temporelles. En effet, les approches "pure MDL" peuvent reposer sur des techniques de compression des données très sophistiquées qui auront un impact important sur les temps de mise à jour / construction du résumé et d'exécution des requêtes.

Afin de découpler autant que faire se peut les deux types de contraintes, nous proposons de limiter la notion de "résumé généraliste" à une structure de données qui laisse l'espace $T \times D$ sous sa représentation native. Un avantage immédiat de cette restriction est que les requêtes s'exécutent sur le résumé de la même façon que sur le flux, ce qui fait disparaître la

contrainte temporelle sur l'exécution des requêtes. Ne restent que les contraintes d'espace-mémoire et de temps de mise à jour / construction.

La discussion qui précède nous amène à proposer la définition suivante pour un "résumé généraliste de flux de données".

Définition : Un résumé généraliste d'un flux de données (tel que défini plus haut) est un algorithme qui maintient une structure de données :

(1) respectant des contraintes d'espace-mémoire et de puissance de calcul,
(2) s'exprimant sous forme de variables appartenant à $T \times D$,
(3) permettant une approximation de toute requête de type SELECT AND COUNT sur $T \times D$,
(4) permettant le calcul de l'erreur d'approximation en fonction des ressources mémoire et CPU disponibles.

Il faut encore spécifier la nature des contraintes. Ainsi, la contrainte sur l'espace-mémoire peut prendre différentes formes selon qu'on souhaite ou non tenir compte du volume du flux écoulé à l'instant courant ($|F(t)|$): la contrainte la plus simple impose une borne fixe au volume du résumé mais on pourrait aussi introduire une borne maximale en $O(\log(|F(t)|)$, voire d'autres dépendances plus complexes (polylog par exemple). Ceci resterait cohérent avec la définition au-dessus : on ne ferait que préciser une classe de résumé généraliste en indiquant le type de contrainte auquel il est soumis. On peut ainsi définir un résumé généraliste à espace constant, à espace logarithmique (en fonction du temps) etc. De la même façon, rien n'interdit d'envisager des contraintes de puissance de calcul évoluant dans le temps.

De façon générale, tout algorithme de flux possède un mode "naturel" de dégradation face à la contrainte temporelle, le sous-échantillonnage aléatoire à réception des événements, dont l'effet sur la précision des réponses aux contraintes SELECT AND COUNT peut être borné ; l'exigence de respect de la contrainte sur la puissance de calcul consiste donc essentiellement à pouvoir produire une borne sur le temps de mise à jour / construction du résumé. La connaissance de cette borne permet de dimensionner a priori le sous-échantillonnage en entrée en fonction des caractéristiques du flux qu'on supposera connues (débit maximal par exemple).

2.3.2 Extensions

La définition ci-dessus a fait disparaître l'exigence d'optimalité pour ne retenir que le respect des contraintes d'espace-mémoire et de puissance de calcul, d'une part, et la capacité à fournir des bornes sur les approximations obtenues, d'autre part.

Nous proposons de réserver le terme de résumé généraliste "optimal" pour des résumés qui reposent sur des modèles de la densité jointe restant en représentation native et minimisant la longueur de description des données (le passé du flux) sous les contraintes d'espace-mémoire (le résumé, c'est-à-dire le modèle, doit tenir en mémoire) et de puissance de calcul.

On peut également étendre la définition au cas distribué : dans ce cas, on considère plusieurs flux F_i et les requêtes portent sur le résumé du flux F qui serait obtenu par multiplexage temporel des flux F_i ($F = MUX(F_i)$). On impose deux propriétés supplémentaires à un résumé généraliste "distribué" :

- il existe un algorithme A permettant de composer les résumés des flux F_i pour former le résumé du flux F: $R(MUX(F_i)) = A(R(F_i))$; cet algorithme est compatible avec les contraintes d'espace-mémoire et de puissance de calcul
- cet algorithme est compatible avec des contraintes de ressources de communication

3 Analyse d'approches de résumés de la littérature

On s'attache dans cette section à comparer certains "résumés" de la littérature à notre définition. On ne discutera pas des qualités et défauts respectifs de ces résumés.

3.1 Echantillonnage à réservoir ("Reservoir sampling")

Cette structure de données maintient à tout instant un échantillon uniforme des événements du flux depuis le début de l'observation du flux (Vitter 1985).

La contrainte (1) de la définition d'un résumé est intrinsèque à la technique : la taille maximale du réservoir est fixée, la mise à jour est en temps constant et réclame moins qu'un tirage aléatoire par événement. La contrainte (2) est évidemment garantie par la technique d'échantillonnage. La contrainte (3) également : les événements présents dans l'échantillon forment un échantillon de la distribution du flux dans $T \times D$. L'application de la théorie des sondages permet d'apporter des bornes à la qualité des approximations en accord avec la contrainte (4).

Cette technique répond donc bien aux critères de notre définition.

Des propositions reposant sur l'échantillonnage à réservoir peuvent hériter de cette propriété: c'est par exemple le cas de l'échantillonnage de la jointure de deux flux reposant sur un double échantillonnage à réservoir pondéré (Féraud *et al.* 2009) ou de l'échantillonnage de flux pour extraire des motifs séquentiels (Raïssi *et al* 2007).

3.2 Echantillonnage progressif : StreamSamp

L'algorithme StreamSamp (Csernel *et al.* 2006) repose également sur la conservation d'échantillons aléatoires des événements du flux. A la différence de l'échantillonnage à réservoir qui ne conserve qu'un seul échantillon, StreamSamp mémorise plusieurs échantillons partitionnant le passé du flux. Tous les échantillons contiennent le même nombre τ d'événements du flux mais couvrent des périodes temporelles de plus en plus longues au fur et à mesure du vieillissement du résumé. En effet, lorsque L échantillons ont été remplis, les deux échantillons les plus anciens sont regroupés et ré-échantillonnés aléatoirement pour constituer un nouvel échantillon dit d'*ordre 1*, de même taille τ, mais dont les éléments sont affectés d'un poids 2. Ce ré-échantillonnage est réalisé récursivement pour former au plus L échantillons de chaque ordre i, chaque élément d'un échantillon d'ordre i étant affecté d'un poids 2^i.

La contrainte (1) est assurée mais avec une croissance logarithmique de la taille du résumé en fonction de la taille du flux. Même si la taille du résumé est potentiellement infinie, la croissance logarithmique est une bonne solution dans la pratique. Les contraintes (2), (3) et (4) sont vérifiées de la même façon que pour l'échantillonnage réservoir à condition de prendre en compte le poids des événements, poids dépendant de l'ordre de l'échantillon dans lequel ils sont conservés.

3.3 Construction de micro-classes : Clustream

La structure de données Clustream (Aggarwal *et al.* 2003) consiste en une série de "clichés" à des instants donnés $C(\delta_n)$ maintenue par décimations successives sur une échelle logarithmique en temps ($\delta_n = \delta*2^n$). Chaque cliché est construit par une adaptation en ligne de l'algorithme de clustering BIRCH, traitant un batch de durée δ. La structure de données reste bien en représentation "native" mais l'algorithme est limité aux données quantitatives.

En conservant l'ensemble de l'historique de fusion / élimination des micro-classes entre deux clichés, et en assimilant l'ensemble des micro-classes à un instant donné à un mélange de gaussiennes (ce qui est cohérent avec leur définition en terme de Cluster Feature Vector, CFV), il est possible d'obtenir une approximation de toute requête SELECT AND COUNT.

Nous n'avons toutefois pas connaissance de bornes sur l'approximation des réponses à ces requêtes.

Chaque cliché contient un nombre maximum de micro-classes décrites chacun par un CFV de taille bornée; par conséquent, l'espace-mémoire nécessaire au maintien de la structure croît au moins en log du temps, comme le nombre de clichés conservés. Nous ne connaissons pas de borne sur la taille de l'historique de fusion / élimination. A ce point près, Clustream respecte une contrainte d'espace-mémoire. Ce dernier point n'est cependant pas forcément anecdotique quand on considère le passé lointain : l'historique de fusion / élimination entre deux clichés peut "peser" bien plus lourd que les clichés eux-mêmes.

Le temps de mise à jour de la structure à l'arrivée d'un événement peut être borné en fonction des caractéristiques du flux (la dimension de l'espace R^p des valeurs, en particulier) et le nombre de micro-classes. Clustream permet donc de respecter une contrainte de puissance de calcul.

Finalement, il apparaît que Clustream s'écarte de notre définition sur la contrainte (4) et éventuellement aussi sur la contrainte (1) (pour l'espace-mémoire nécessaire).

3.4 Extraction de motifs fréquents

Ce type de résumé s'adresse seulement à des données qualitatives : les données associées à un événement sont représentées par un multiset de modalités de tout ou partie des variables. Le treillis des motifs ou itemsets (et les comptes associés) peut être considéré comme une représentation hiérarchique de la table de Burt (ensemble des comptes associés aux éléments du produit cartésien de la représentation disjonctive complète des variables sur leurs modalité) qui est l'équivalent pour les données qualitatives de la notion de "densité".

Dans un contexte de bases de données, l'ensemble des motifs fréquents peut donc être considéré comme une approximation de la table de Burt, donc de la densité sur l'espace de description des événements.

Dans un contexte de flux de données, le traitement de la dimension temporelle en fenêtres batch permet de définir une structure de données respectant la structure native des données et permettant de répondre de façon approximative à toute requête SELECT and COUNT. L'agrégation des comptes de deux fenêtres adjacentes pose néanmoins une difficulté majeure : un motif qui n'est fréquent dans aucune des deux fenêtres ne peut être fréquent dans l'union de ces deux fenêtres et l'agrégation se fait donc "vers le haut" (au sens de l'ordre partiel défini sur les motifs par l'inclusion) à partir des motifs fréquents déjà connus. La difficulté provient de ce que dans cette agrégation, certains comptes ne sont pas connus dans les résumés de départ (ceux des motifs qui n'étaient pas fréquents). Cela induit inéluctablement

que des motifs non fréquents mais qui le deviendront après une plus longue observation du flux, ne pourront pas être observés. D'où l'existence de motifs faux négatifs et des résultats en rappel dégradés. Par ailleurs, l'agrégation des comptes de deux fenêtres adjacentes peut ne fournir qu'une borne inférieure du compte réel sur l'union des deux fenêtres.

On peut citer à titre d'exemple, les algorithmes Fiasco (Symphor et al. 2008) et Spams (Vinceslas et al. 2009), qui apportent une réponse satisfaisante à cette difficulté dans le cas de fenêtres définies depuis le début de l'observation du flux. Ces algorithmes reposent sur la construction incrémentale d'une structure d'automate qui permet l'indexation des motifs fréquents du flux de données. Le traitement du flux s'effectue par fenêtres batch sans contrainte particulière sur la taille de celles-ci qui peuvent être réduites à une transaction. Il n'est point également nécessaire de supposer, à l'exécution de ces algorithmes, la connaissance de l'alphabet des items ainsi que la connaissance du nombre des id du flux. Cette information est apprise de façon incrémentale, à la volée, au fur et à mesure de l'insertion des nouvelles fenêtres batch du flux.

Afin d'améliorer les réponses en rappel, ainsi que l'agrégation des comptes de fenêtres adjacentes, les approches Fiasco et Spams proposent de biaiser la valeur du support. En effet, au lieu de n'indexer que les seuls motifs θ fréquents, elles indexent plus largement tous les motifs $(\theta - \varepsilon)$ fréquents conservant aussi des motifs non fréquents susceptibles de devenir fréquents. Le choix de ε résulte des travaux sur les couvertures statistiques (Laur et al. 2007) qui permettent de conserver le nombre minimal de motifs supplémentaires et d'obtenir une approximation satisfaisante. En réponse à une requête SELECT AND COUNT, il n'y a pas de motifs faux négatifs et le rappel est égal à 1 avec une forte probabilité. Le compte des motifs fréquents n'est plus seulement une borne inférieure mais un compte exact.

La contrainte sur la taille de ce type de résumé est très faible, seulement linéaire et nous n'avons pas connaissance de borne non triviale sur le temps d'exécution d'une recherche de motifs fréquents. Le recours aux fenêtres logarithmiques semble offrir un renforcement naturel de la contrainte sur la taille du résumé.

3.5 Cubes de données et flux

Ce type de résumé s'applique seulement à des données multidimensionnelles : la description associée à un événement est telle qu'il existe une application de Q (les dimensions d'analyse) dans R^p (les mesures). La sémantique d'une description associée à un événement est alors du type "le produit X dans le magasin Y a été acheté Z fois à l'instant t (où t est l'estampille temporelle de l'événement)". Les variables qualitatives Q servent à structurer un cuboïde. Au fur et à mesure que les données arrivent, on met à jour le cuboïde en agrégeant les mesures batch par batch.

Afin de satisfaire la contrainte d'espace-mémoire, la dimension temporelle est compressée dans des fenêtres logarithmiques. L'historique d'une cellule (un point de R^p) est conservé à une faible granularité temporelle pour les événements récents mais cette précision se dégrade à mesure que l'événement vieillit. De plus, il est fréquent que les variables qualitatives puissent être considérées à différents niveaux de granularité. On appelle hiérarchie cette succession de niveaux. En pratique, un utilisateur consultant un résumé de flux s'intéresse aux données précises pour les événements récents et observe les tendances (i.e. les données à une plus faible granularité) pour les données plus anciennes. Les auteurs de (Pitarch *et al* 2008) appliquent donc le mécanisme des fenêtres logarithmiques sur toutes les dimensions hiérarchisées et déterminent a priori (grâce aux *fonctions de précision* définies en fonction

des requêtes utilisateurs) quand un niveau de granularité ne sera plus consulté. Par exemple, les ventes précises par *produit* peuvent être conservées pendant un mois puis, dans la mesure où cette précision n'est plus demandée par la suite, ces données sont agrégées au niveau *catégorie de produit*.

En ce qui concerne la contrainte (1), comme pour l'extraction de motifs fréquents, l'utilisation de fenêtres logarithmiques permet de contraindre la taille du résumé. Il est cependant intéressant de noter que la difficulté rencontrée par l'extraction de motifs fréquents concernant l'union de fenêtres n'existe pas ici. Chaque fenêtre contient le nombre exact d'apparitions d'un item dans l'intervalle de temps concerné. L'agrégation de deux fenêtres adjacentes fournit donc le compte exact sur l'union des deux fenêtres. Toute fenêtre résume l'ensemble de $T \times D$ sur l'intervalle de temps associé à la fenêtre : la contrainte (2) est respectée. Par définition même d'un cube de données, la contrainte (3) est respectée. Il faut cependant noter que la qualité de l'approximation est dépendante de la partie du résumé interrogé (plus cette partie est vieille, plus l'approximation sera grossière). Finalement, il apparaît que cette approche s'écarte légèrement de notre définition car il est possible de déterminer des bornes concernant la qualité de l'approximation produite grâce aux *fonctions de précision* et non en fonction des ressources mémoire et CPU disponibles.

3.6 Algorithme REGLO

L'algorithme REGLO (Marascu 2009) considère un flux d'événements de type numérique de dimension p ($D = R^p$). Chaque dimension correspond par exemple à un capteur dont on observe une série temporelle. Si on note n le nombre d'événements qui sont arrivés depuis le début du flux, la taille de l'historique du flux complet est $n*(p+1)$ si on conserve aussi l'estampille temporelle. L'algorithme REGLO propose de limiter l'espace de stockage du flux à une valeur s fixe ($s << n*(p+1)$) et d'optimiser les valeurs conservées par un partage non égalitaire de l'espace de stockage entre les séries temporelles. Ce partage est basé sur l'idée que les séries minimisant l'erreur de reconstruction de leurs données d'origine nécessitent moins d'espace que les séries présentant une erreur élevée. Ainsi, REGLO n'impose plus la conservation des événements complets mais de données partielles pour chaque événement. La sélection des valeurs conservées se fait sous une contrainte de borne sur l'erreur entre la vraie valeur et la valeur interpolée entre 2 valeurs conservées d'une série.

La contrainte (1) est donc bien vérifiée par la limite s. La contrainte (2) n'est pas respectée strictement mais il est facile de reconstituer par interpolation des événements complets dans $T \times D$. La contrainte (3) est vérifiée pour la même raison que la contrainte (2). La contrainte (4) est également respectée par l'algorithme de choix des valeurs conservées pour chaque série.

4 Analyse du traitement de la dimension temporelle

L'estimation de densité sur $T \times D$ peut se faire : (1) en considérant globalement l'espace $T \times D$; (2) en découplant la dimension temporelle dans l'estimation, par la définition d'un système de *fenêtres temporelles* au sein desquelles on réalise une estimation de la densité sur D (éventuellement sur $T_f \times D$ où T_f est la partie de T correspondant à la fenêtre). La caractéristique majeure de l'approche (2) est que la structure des fenêtres temporelles est connue à

l'avance indépendamment des données, ce qui permet la mise en place d'algorithmes plus efficaces pour la construction et la mise à jour du résumé mais au détriment de l'optimalité de la représentation du résumé. Afin de permettre la reconstitution d'un résumé sur n'importe quelle période du passé, il est nécessaire de pouvoir construire l'estimation de la densité de l'union de deux fenêtres temporelles à partir du résumé des deux fenêtres (et non du flux détaillé).

Dans les approches présentées précédemment, trois systèmes de fenêtrage sont considérés :

- Une *fenêtre unique* couvrant toute la période d'observation du flux jusqu'à l'observation de l'instant présent.
- Des fenêtres dites de type « *batch* » : il s'agit de fenêtres adjacentes de taille fixe, en général définies au niveau logique (nombre d'événements) plutôt qu'au niveau physique afin de maîtriser leur taille.
- Des fenêtres dites à structure *logarithmique* (aussi appelées *tilted time windows*) dont la couverture temporelle varie de façon exponentielle au fur et à mesure qu'elles vieillissent.

Le tableau ci-dessous synthétise notre analyse des différentes approches de résumé suivant ce critère de traitement de la dimension temporelle.

Approche de résumé	**Type de fenêtres / taille du résumé**	**Découplage de la dimension temporelle**	**Méthode d'estimation de la densité sur D**	**Estimation de la densité sur l'union de 2 fenêtres**
Réservoir	Fenêtre unique / fixe	Non	Echantillonnage aléatoire	Sans objet
StreamSamp	Logarithmique / logarithmique	Oui	Echantillonnage aléatoire	Oui : ré-échantillonnage aléatoire
Clustream	Logarithmique / logarithmique au mieux	Partiel	Par soustraction des clichés correspondant aux bornes de la fenêtre	Oui : soustraction des clichés correspondant aux bornes de la fenêtre
Motifs fréquents	Batch / linéaire	Oui	Co-occurrences d'items au-dessus du support	Non, ou de façon approximative
Cuboïdes	Batch / logarithmique	Non	Agrégation multi-dimensionnelle	Oui : agrégation multi-dimensionnelle
REGLO	Fenêtre unique / fixe	Non	Coefficients de régression	Sans objet

L'approche Clustream réalise un découplage partiel de la dimension temporelle. En effet, l'introduction de la dimension temporelle dans le CFV permet de conserver une estimation globale sur $T \times D$ de la densité, cette densité étant représentée par un ensemble de micro-classes. La conservation des clichés permet de construire une estimation de la densité globale pour n'importe quelle fenêtre du passé, à condition que des clichés aient été conservés correspondant aux bornes de ces fenêtres. Ainsi, il n'est pas nécessaire de disposer d'un algo-

rithme d'estimation de la densité de l'union de deux fenêtres adjacentes. Cette approche peut sembler idéale mais se heurte au problème de la qualité de l'estimation globale lorsque celle-ci est réalisée par simple ajout de la dimension temporelle aux données du flux.

Les cuboïdes sont alimentés en considérant des fenêtres de type batch traitées de façon consécutive indépendamment les unes des autres. Il peut donc sembler en première analyse qu'il y a un découplage de la dimension temporelle. En réalité, l'estimation de densité s'opère bien sur toute la dimension temporelle car des hiérarchies sur d'autres dimensions que le temps peuvent être utilisées pour agréger les données anciennes lorsque le cube occupe trop de place. Ainsi la dimension temporelle permet de piloter l'oubli des données (plus les données sont anciennes, moins il y a de détail) mais les agrégations réalisées pour l'oubli ne portent pas uniquement sur la dimension temporelle. Cet aspect est très intéressant : il est rendu possible par une définition a priori des domaines des dimensions et des hiérarchies associées.

5 Construction et utilisation des résumés de flux de données

La gestion de résumés de flux de données pose deux types de problèmes : (1) comment sont construits les résumés ; (2) comment sont utilisés les résumés.

La construction des résumés doit clairement être réalisée au sein du moteur d'un SGFD. En effet, cette construction devant être faite à la volée, elle bénéficie naturellement de l'infrastructure d'un SGFD. Pour chaque méthode de résumé, il y a tout intérêt à reporter au maximum les traitements sur des fonctionnalités de base du SGFD, qui pourra donc optimiser l'espace de stockage et le temps CPU en relation avec les autres requêtes traitées. Le stockage des résumés est un stockage permanent sur disque, soit dans des fichiers soit dans une base de données.

L'utilisation des résumés peut répondre à des besoins très divers, dont nous donnons ci-dessous les deux classes principales :

- Répondre à des requêtes portant sur le passé du flux, à l'exception de données récentes. Le paradigme standard de l'interrogation des bases de données est conservé (requêtes one-shot). Il est intéressant de laisser à l'utilisateur la possibilité d'exprimer sa requête sur le schéma du flux comme si celui-ci était complètement disponible, mais de répondre en traduisant cette requête en une requête sur le résumé et en produisant un intervalle de confiance sur le résultat. Un exemple d'une telle requête est de calculer la consommation électrique totale d'une région au mois de Janvier 2007. Lorsque le résumé est un échantillon aléatoire, le résultat est assorti d'un intervalle de confiance calculé à l'aide de la théorie des sondages.
- Répondre à des requêtes continues portant à la fois sur des parties anciennes du flux et sur le contenu actuel flux. Dans ce cas, les requêtes relèvent du paradigme des SGFD. Un exemple d'une telle requête est de produire une alarme lorsque la consommation électrique d'une région dépasse de 10% la consommation de la même région une année auparavant. Il faut dans ce cas savoir synchroniser les données actuelles du flux avec celles mémorisées dans les résumés. Il faut également pouvoir exécuter efficacement des requêtes continues 'glissantes' sur les résumés, paradigme intermédiaire entre la gestion de bases de données et de flux de données.

Tous ces aspects liés à la construction et à l'utilisation des résumés de flux de données sont étudiés dans le projet collaboratif MIDAS (MIDAS 2008-2010), où une plate-forme de démonstration est en cours de développement dans l'environnement du SGFD Esper (cf. MIDAS).

6 Conclusion

Dans ce qui précède, nous avons cherché à préciser la notion de "résumé généraliste" dans un contexte de flux de données et à en proposer une définition réaliste.

Un résumé généraliste est défini comme une structure de données permettant une approximation contrôlée de toute requête de type SELECT and COUNT sur le temps et la description des événements, sous contrainte de ressources mémoire et de puissance de calcul.

En comparant certains "résumés" de la littérature à cette définition, nous avons montré comment cette définition peut constituer une base pragmatique pour la comparaison de ces différentes structures de données.

Il reste encore de nombreuses formes de structures de données présentées dans la littérature comme des "résumés" de flux de données : motifs temporels fréquents, automates etc. Nous nous attacherons dans la suite de ce travail à les confronter à notre définition.

Références

Adriaans P., Vitanyi P., The Power and Perils of MDL, IEEE International Symposium on Information Theory, 2007. ISIT 2007, pp. 2216-2220.

Aggarwal C.C., Han J., Wang J., Yu P.S., A Framework for clustering evolving data streams, Conférence VLDB, 2004.

Cormode G., Muthukrishnan S., An improved data stream summary: The count-min sketch and its applications. In Proceedings of Latin American Theoretical Informatics (LATIN), pages 29-38, 2004.

Csernel B., Clérot F., Hébrail G., StreamSamp: Data stream clustering over tilted windows through sampling, Workshop ECML/PKDD Knowledge Discovery from Data Streams, Berlin, Sept.2006.

Esper, Open Source Complex Event Processing System, http://esper.codehaus.org/ , 2009.

Féraud R, Clérot F, Gouzien P, Sampling the join of streams, IFCS'2009 Int. Conf. on Int. Federation of Classification Societies, Dresden, 2009.

Flajolet P., Martin G.N., Probabilistic Counting Algorithms for Data Base Applications, In Journal of Computer and System Sciences, Vol. 31(2), 1985, pp. 182-209.

Golab L., Özsu M.T., Issues in Data Stream Management. SIGMOD Record, Vol. 32, No. 2, June 2003.

Grünwald P.D., The Minimum Description Length Principle. MIT Press, June 2007.

Laur P.A., Nock R., Symphor J.E.., Poncelet P. : Mining evolving data streams for frequent patterns. Pattern Recognition 40 (2): 492-503 (2007)

Marascu A., Extraction de motifs séquentiels dans les flux de données, thèse de doctorat, Université de Nice-Sophia Antipolis, septembre 2009.

MIDAS, Projet MIDAS, ANR-MDCO-008, http://midas.enst.fr , 2008-2010.

Pitarch Y., Laurent A., Plantevit M., Poncelet P.: Fenêtres sur cube. BDA 2008, Guilherand Granges, France. pp. 1-20.

Quinlan, J. R., C4.5: Programs for Machine Learning. San Mateo: Morgan Kaufmann, 1993.

Raïssi C., Poncelet P.: Sampling for Sequential Pattern Mining: From Static Databases to Data Stream. Proceedings of the IEEE International Conference on Data Mining (ICDM 07), Omaha NB, USA, October 2007

Symphor J.E., Mancheron A.,Vinceslas L., Poncelet P.: Le FIA: un nouvel automate permettant l'extraction efficace d'itemsets fréquents dans les flots de données, EGC 2008: 157-168

Vinceslas L.,Symphor J.E., Mancheron A.,Poncelet P.: SPAMS: Une nouvelle approche incrémentale pour l'extraction de motifs séquentiels fréquents dans les data streams.EGC 2009 : 205-216.

Vitter, J. S., Random Sampling with a Reservoir. ACM Transactions on Mathematical Software, Vol. 11, No. 1, March 1985. Pages 37-57.

Summary

When the volume of data is too high to be stored in a database at a reasonable cost or when data is arriving at a high speed, Data Stream Management Systems (DSMS's) can capture streams of structured records and query them on the fly by defining permanent queries (executed continuously). But DSMS's do not save the history of the streams which is lost forever. This paper proposes a formal definition of what should be a generic summary of the history of a structured data stream. Generic refers here to the possibility of answering various queries or performing various mining tasks from the summary instead of the whole stream. Then it reviews several summarizing approaches according to this definition. Finally the integration of such data stream summarizing approaches in a DSMS is discussed, both in terms of construction and use.