Délestage pour l'analyse multidimensionnelle de flux de données

Sylvain Ferrandiz, Georges Hébrail

GET / Télécom Paris
46, rue Barrault
F-75634 Paris Cedex 13
sylvain.ferrandiz@enst.fr
georges.hebrail@enst.fr

Résumé. Dans le contexte de la gestion de flux de données, les données entrent dans le système à leur rythme. Des mécanismes de délestage sont à mettre en place pour qu'un tel système puisse faire face aux situations où le débit des données dépasse ses capacités de traitement. Le lien entre réduction de la charge et dégradation de la qualité des résultats doit alors être quantifié.

Dans cet article, nous nous plaçons dans le cas où le système est un cube de données, dont la structure est connue a priori, alimenté par un flux de données. Nous proposons un mécanisme de délestage pour les situations de surcharge et quantifions la dégradation de la qualité des résultats dans les cellules du cube. Nous exploitons l'inégalité de Hoeffding pour obtenir une borne probabiliste sur l'écart entre la valeur attendue et la valeur estimée.

1 La gestion de flux de données

Les avancées de l'électronique et de l'informatique enrichissent continuellement la pratique de la récolte et de la gestion des données. La constante est l'accroissement des capacités de traitement, tant au niveau de l'acquisition que du stockage et de l'accès aux données. Mais lorsque l'information doit être extraite instantanément de données récoltées continuellement, le modèle relationnel basé sur des tables atteint ses limites. C'est là qu'interviennent les flux de données.

Un flux de données est une suite de tuples ayant tous la même structure. Cette structure est représentée par un schéma, comprenant le nom des champs du tuple et leur type. La différence entre un flux et une table est le caractère ordonné des tuples. L'ordre est souvent déterminé par un champ d'agencement (typiquement la date, mais pas nécessairement). On entre dans le cadre de la gestion de flux dès lors que

- les données du flux n'ont pas vocation à être stockées,
- les données nécessitent un traitement immédiat,
- les requêtes sont exécutées continuellement (*i.e.* les flux de données donnent naissance à d'autres flux de données).

La gestion de flux de données repose sur un modèle "data push" : les données se présentent d'elles-mêmes, à leur propre rythme. En conséquence, le système ne maîtrise pas et ne connaît

pas à l'avance le rythme et ses fluctuations. Des mécanismes d'adaptation sont à mettre en place pour faire face à toutes les situations, notamment lorsque le rythme s'accélère au point que le système ne peut plus traiter instantanément toutes les données. Dans ce cas, un mécanisme de réduction de la charge procède à une élimination des tuples en différents points du processus de traitement. On parle de mécanisme de délestage (ou : load shedding). Le système entre alors dans un mode "dégradé" et il convient de mesurer l'impact du délestage sur les critères de qualité.

Nous étudions ici un tel mécanisme dans le contexte suivant : le système est un cube de données alimenté par un flux de données. On parle à ce sujet de stream cubing (Han et al. (2005)). La structure du cube est supposée connue a priori et ne fluctue pas avec les données. Notre apport réside dans la proposition d'une mesure probabiliste de l'erreur commise sur le calcul de la valeur moyenne dans chacune des cellules du cube.

L'article est composé comme suit. Dans la section 2, nous discutons certains mécanismes de délestage. Dans la section 3, nous obtenons une borne probabiliste sur l'erreur commise suite à un échantillonnage différencié. Dans la section 4, nous montrons l'intérêt de l'usage de cette borne probabiliste pour estimer la perte de qualité du résultat dans chaque cellule du cube. Enfin, nous concluons à la section 5.

2 Le délestage

Afin d'assurer le fonctionnement du système lorsque le débit des flux dépasse la capacité de traitement du système, un mécanisme pertinent d'oubli des tuples permet de réduire la charge : on parle de délestage (Motwani et al. (2003)). La plupart des travaux sur le délestage portent sur les systèmes de gestion de flux de données (Babcock et al. (2002), Stonebraker et al. (2005), Carney et al. (2002)).

Dans le cadre du projet Aurora (Abadi et al. (2003), un mécanisme de délestage a été proposé. Il exploite les notions d'introspection et de qualité de service (Tatbul et al. (2003)). L'introspection permet d'estimer le temps de traitement global d'un tuple entrant dans le système et la qualité de service (qui se dégrade avec l'allongement du temps de traitement) permet de déterminer le passage à un mode dégradé. La réduction de charge est équilibrée par la qualité de service. Notons que c'est l'administrateur du système qui spécifie les courbes de qualité de service.

Dans le contexte de la classification supervisée, Chi et al. (2005) considèrent un classifieur alimenté par un flux de données. Ils proposent d'utiliser le taux d'erreur du classifieur pour guider la réduction de charge et l'importance d'un tuple est mesurée par son influence sur le taux d'erreur du classifieur. La réduction de charge est ainsi équilibrée par la qualité prédictive d'un classifieur.

En se restreignant aux opérateurs d'agrégation, comme les opérateurs de calcul de moyenne, de minimum, de médiane, Babcock et al. (2004) proposent l'erreur de calcul comme facteur d'équilibre du délestage. Dans ce sens, l'élimination des tuples est contrôlée par la dégradation des résultats des opérateurs d'agrégation. Les auteurs proposent d'utiliser une approche probabiliste et la borne de Hoeffding.

3 Liaison entre erreur et taux d'échantillonnage

Afin d'étudier l'impact de la réduction de la charge sur la qualité du résultat dans chaque cellule du cube, il nous faut lier taux d'échantillonnage et erreur sur le calcul de la moyenne. Nous exploitons pour cela l'inégalité de Hoeffding. Pour le confort du lecteur, nous présentons notre résultat indépendamment de son application. Notons que notre application de la borne de Hoeffding différe de celle de Babcock et al. (2004). Il s'agit ici de formuler une borne sur l'erreur d'estimation d'une moyenne à partir de données échantillonnées suivant des taux d'échantillonnage différents.

Soit N un entier et soient $M_1, \ldots, M_N$ des variables aléatoires indépendantes deux à deux. Soient $B_1, \ldots, B_N$ des variables de Bernoulli indépendantes deux à deux et indépendantes des M_n. Ces variables sont réparties dans L paquets, les N_l variables du l^{eme} paquet suivant une loi de Bernoulli de paramètre p_l.

Notons $Y_n = M_n B_n$. Si $M_n \in [a_n, b_n]$ est bornée pour tout n, l'inégalité de Hoeffding (1963) pour les Y_n s'écrit, pour tout $\varepsilon > 0$,

$$P\left(\left|\sum_n Y_n - \mathbb{E}(\sum_n Y_n)\right| \geq \frac{N\varepsilon}{2}\right) \leq 2\exp\left(-\frac{N^2\varepsilon^2}{2\sum_n (b_n - a_n)^2}\right). \tag{1}$$

Si on suppose les M_n toutes de même espérance μ et $b_n - a_n = \lambda$ pour tout n, en notant $p = \sum_l p_l N_l$, on peut réécrire l'inégalité comme suit

$$P\left(\left|\frac{1}{p}\sum_n Y_n - \mu\right| \geq \frac{N\varepsilon}{2p}\right) \leq 2\exp\left(-\frac{N\varepsilon^2}{2\lambda^2}\right). \tag{2}$$

En utilisant l'inégalité triangulaire, l'additivité des probabilités et l'inégalité de Hoeffding par deux fois (pour les Y_n et pour les M_n), on obtient les majorations

$$\begin{aligned}
& P\left(\left|\frac{1}{p}\sum_n Y_n - \frac{1}{N}\sum_n M_n\right| \geq \frac{N\varepsilon}{p}\right) \\
\leq\ & P\left(\left|\frac{1}{p}\sum_n Y_n - \mu\right| \geq \frac{N\varepsilon}{2p}\right) + P\left(\left|\frac{1}{N}\sum_n M_n - \mu\right| \geq \frac{N\varepsilon}{2p}\right) \\
\leq\ & 2\exp\left(-\frac{N\varepsilon^2}{2\lambda^2}\right) + 2\exp\left(-\frac{N^3\varepsilon^2}{2p^2\lambda^2}\right) \\
\leq\ & 4\exp\left(-\frac{N\varepsilon^2}{2\lambda^2}\right) \text{ (car } N/p \geq 1\text{).}
\end{aligned} \tag{3}$$

On déduit de cette inégalité que, sous les hypothèses données, on commet une erreur inférieure à $\eta = \frac{N\varepsilon}{p}$ avec une probabilité supérieure à $1 - \beta$ en remplaçant $\frac{1}{N}\sum_n M_n$ par $\frac{1}{p}\sum_n M_n B_n$, avec $\beta = 4\exp\left(-\frac{N\varepsilon^2}{2\lambda^2}\right)$. D'où la relation

$$\eta = \frac{\lambda\sqrt{2N}}{p}\sqrt{\log\frac{4}{\beta}}. \tag{4}$$

4 Notre proposition

Nous proposons maintenant un mécanisme de délestage pour gérer la surcharge d'un système composé d'un cube de données alimenté par un flux de données. Nous montrons comment le résultat de la section 3 conduit à une caractérisation de la dégradation de la qualité du cube suite à l'échantillonnage de données.

Contexte d'application. – La structure du cube est supposée connue a priori et fixe : les dimensions ne varient pas en nombre et en structure au cours du temps. Les cellules du cube considéré sont celles correspondant au niveau de granularité le plus fin. Nous supposons que le cube possède une dimension temporelle. Son domaine est supposé découpé en périodes toutes de même longueur T. C'est un paramètre orienté usage caractérisant le niveau d'observation temporelle le plus fin et on peut imaginer que T correspond à une journée et vaut 24h.

Le flux de données entrant possède le schéma $(\mathcal{D}_1, \ldots, \mathcal{D}_u, \mathcal{M}_1, \ldots, \mathcal{M}_v, \mathcal{T})$. Un tuple est alors un vecteur $(d_1, \ldots, d_u, m_1, \ldots, m_v, t)$, où les u premiers champs servent à calculer les dimensions du cube, les v suivants sont les mesures d'intérêt et t est une estampille temporelle. Les domaines de chacun des v attributs de mesure sont supposés bornés, de tailles respectives $\lambda_1, \ldots, \lambda_v$.

Mise en œuvre. – Définissons maintenant la politique d'échantillonnage. D'une part, nous proposons de traiter les tuples entrants par buffers successifs et disjoints. Notons, pour un buffer l, N_l le nombre de tuples et δ_l la durée de constitution du buffer (le délai entre l'arrivée du tuple le plus ancien et l'arrivée du tuple le plus récent). Notons τ le temps de traitement d'un tuple. Si p_l est le taux d'échantillonnage dans le buffer l, la contrainte sur la charge est la suivante :

$$p_l N_l \tau \leq \delta_l. \tag{5}$$

Autrement dit, on souhaite que le délai de traitement global des tuples qui seront conservés ne dépasse pas le délai d'acquisition des tuples du paquet. On obtient la probabilité d'échantillonnage p_l dans le buffer l : $p_l = \frac{\delta_l}{N_l \tau}$. Dès lors, chaque tuple du buffer l est indépendamment soumis à un tirage aléatoire avec probabilité p_l.

Nous proposons deux stratégies pour fixer la taille des buffers : fixer la taille des buffers de manière explicite (*i.e.* en posant $N_l = N$ pour tout l) ou implicite (*i.e.* en posant $\delta_l = \delta$ pour tout l). La présentation qui suit est indifférente au choix de stratégie et nous adoptons arbitrairement ici la seconde.

Caractérisation de l'erreur. – Considérons une cellule c du cube à son niveau le plus fin. Cette cellule contient un ensemble de vecteurs de mesures $(m_1, \ldots, m_v)$. Ces mesures sont relatives à un certain nombre de caractéristiques dimensionnelles, notamment la caractéristique temporelle : elles relèvent toutes de la même période de longueur T. Pour simplifier la description, supposons $v = 1$.

La valeur d'intérêt pour chaque cellule est la moyenne $\overline{m_1^{(c)}}$ des $N^{(c)}$ mesures qu'elle contiendrait si les tuples n'avait subi aucun échantillonnage en entrée du système. Les mesures contenues dans la cellule résultent en fait d'un échantillonnage par paquets, avec des taux d'échantillonnage p_l différents dans chacun des L paquets. Notons $N_l^{(c)}$ le nombre de tuples du buffer l dont les mesures tombent dans la cellule c (on suppose ce nombre calculable en un

temps négligeable, au moins inférieur à δ). C'est la valeur moyenne $\widehat{m_1^{(c)}}$ des valeurs échantillonnées dont on dispose. Celle-ci est calculée comme la somme des valeurs échantillonées divisée par $\sum_l p_l N_l^{(c)}$, en accord avec le résultat obtenu à la section 3.

En combinant la relation (4) avec l'équation de charge (5), on est capable d'affirmer avec une probabilité supérieure à $1 - \beta$ que l'erreur commise en considérant $\widehat{m_1^{(c)}}$ et non $\overline{m_1^{(c)}}$ est

$$\eta = \frac{\lambda\sqrt{2N^{(c)}}}{\sum_l \frac{N_l^{(c)}}{N_l}} \frac{\tau}{\delta} \sqrt{\log \frac{4}{\beta}}. \tag{6}$$

Propriété. – Nous avons jusqu'ici considéré le niveau le plus fin du cube. On obtient plus généralement une estimation de l'erreur sur toute agrégation de données contenues dans plusieurs cellules. Deux cas sont à distinguer successivement.

Pour deux (ou plusieurs) cellules dont les données correspondent à une même période de longueur T, la borne se calcule à partir des nombres de tuples tombant dans le groupe de cellules avant et après échantillonnage. Pour une agrégation sur deux (ou plusieurs) périodes, la borne se calcule à partir de l'ensemble des buffers constitué au cours des différentes périodes.

5 Conclusion

De nombreuses applications nécessitent le traitement continuel de données qui se présentent à leur propre rythme. Les systèmes ayant à gérer de tels flux de données doivent intégrer des mécanismes de gestion de la surcharge.

Dans cet article, nous avons proposé un mécanisme de délestage pour un système composé d'un cube de données alimenté par un flux de données. La nouveauté majeure consiste en l'exploitation de l'inégalité de Hoeffding pour quantifier l'erreur sur le résultat agrégé dans chaque cellule du cube.

Il reste à passer en phase expérimentale. Notamment, il faut résoudre la question de l'estimation du temps de traitement moyen d'un tuple. Plusieurs politiques existent : calculer un temps moyen absolu, considérer le temps de traitement moyen sur le paquet précédent, opérer une modélisation auto-régressive du temps de traitement, employer un modèle de régression liant le temps de traitement à différentes caractéristiques du système. Seule la pratique permettra de trancher entre ces différentes politiques.

Références

Abadi, D. J., D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, et S. Zdonik (2003). Aurora : a new model and architecture for data stream management. *The VLDB Journal 12*(2), pp. 120–139.

Babcock, B., S. Babu, M. Datar, R. Motwani, et J. Widom (2002). Models and issues in data stream systems. In *PODS '02 : proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems*, New York, NY, USA, pp. 1–16. ACM Press.

Babcock, B., M. Datar, et R. Motwani (2004). Load shedding for aggregation queries over data streams. In *ICDE'04 : proceedings of the 20th international conference on data engineering*, Washington, DC, USA, pp. 350–361. IEEE Computer Society.

Carney, D., U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, et S. Zdonik (2002). Monitoring Streams - A New Class of Data Management Applications. In *VLDB'02 : proceedings of the twenty-eighth international conference on very large data bases*, Hong Kong, China, pp. 215–226.

Chi, Y., H. Wang, et P. Yu (2005). Loadstar : load shedding in data stream mining. In *VLDB '05 : proceedings of the 31st international conference on very large data bases*, pp. 1302–1305. VLDB Endowment.

Han, J., Y. Chen, G. Dong, J. Pei, B. W. Wah, J. Wang, et Y. D. Cai (2005). Stream cube : An architecture for multi-dimensional analysis of data streams. *Distrib. Parallel Databases 18*(2), pp. 173–197.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*(301), pp. 3–30.

Motwani, R., J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. S. Manku, C. Olston, J. Rosenstein, et R. Varma (2003). Query processing, approximation, and resource management in a data stream management system. In *CIDR*.

Stonebraker, M., U. Cetintemel, et S. Zdonik (2005). The 8 requirements of real-time stream processing. *SIGMOD Rec. 34*(4), pp. 42–47.

Tatbul, N., U. Çetintemel, S. B. Zdonik, M. Cherniack, et M. Stonebraker (2003). Load shedding in a data stream manager. In *VLDB*, pp. 309–320.

Summary

In the context of data stream management, incoming data feed the system at their own rate. In situation when the data flow overheads system capacity, load shedding mechanisms aim at reducing system load by dropping tuples cleverly. The effect of the reduction of the system load on the degradation of the quality of the system output has to be quantified.

In this article, we consider that the system is an OLAP data cube, which structure is a priori known and fed with a data stream. We give a load shedding scheme and propose a probabilistic evaluation of the error made in each cell of the cube. We make use of the Hoeffding bound in order to derive a probabilistic bound on the error based on the expected value and the estimated value.