Vers la prise en considération des périodes d'instabilité dans les SGBD temps réel

Claude Duvallet

LIH, Faculté des Sciences et Techniques, 25 rue Philippe Lebon, BP 540,
F-76058 LE HAVRE CEDEX*, Courriel : Claude.Duvallet@univ-lehavre.fr*

Résumé – *Au cours de ces dernières années, les applications informatiques sont devenues beaucoup plus exigeantes en termes de services rendus aux utilisateurs et de données manipulées. De plus en plus de ces applications manipulent des données en quantités très importantes et les traitements effectués sur ces données sont sujets à des contraintes temporelles. Traditionnellement, ces applications étaient gérées par des systèmes temps réel, bien adaptés pour la gestion des contraintes temporelles. Mais, ces systèmes sont beaucoup moins efficaces pour manipuler des quantités importantes de données. Les SGBD temps réel sont donc nés de la nécessité de gérer des données en grandes quantités tout en respectant des contraintes temporelles pour le traitement de ces données.*

Mots-clés – *SGBD temps réel, transactions temps réel, données temps réel, contrôle par rétroaction.*

I INTRODUCTION

Les applications informatiques deviennent de plus en plus sophistiquées en termes de données. Certaines de ces applications utilisent des données qui ne sont valides que durant une période assez courte, on parle alors de données temps réel. De plus, certaines de ces applications, pour répondre aux besoin des utilisateurs, doivent respecter des contraintes temporelles. Les SGBD temps réel doivent permettre de prendre en compte la manipulation de volumes important de données (parfois temps réel) tout en respectant les contraintes temporelles des applications [6, 11].

Parmi ces applications, on peut citer les applications sur lesquelles reposent la surveillance de sites industriels à très haut risque (usines chimiques, centrales nucléaires, etc.). De nombreux capteurs (température, pression, etc.) sont alors utilisés et permettent de connaître en permanence l'état du site industriel. Ils servent à mettre à jour périodiquement la base de données temps réel qui reflète l'état du site (appelé souvent état du monde réel). Si les données ne sont pas mises à jour régulièrement, elles ne peuvent plus refléter l'état réel du site industriel surveillé à un instant ***t***. Bien évidement, ces données sont utilisés pour détecter d'éventuels dysfonctionnements. Les traitements qui permettent de les détecter dans le cadre d'un SGBD sont basés sur des transactions plus ou moins complexes. Pour que l'utilisateur soit averti suffisamment tôt d'un dysfonctionnement ou d'un incident, le processus qui mène à cette détection doit être effectué en respectant certaines contraintes temporelles, d'où l'utilisation de transactions temps réel.

Les travaux sur les SGBD temps réel (SGBDTR) ont débuté en 1988 [1]. Ces travaux concernaient pour beaucoup la gestion du contrôle de concurrence et l'ordonnancement temps réel [11]. De nombreux protocoles de contrôle de concurrence ont été conçus pour la gestion des transactions temps réel et de plus amples détails peuvent être trouvés dans [6]. Dans cet article, nous allons commencer par présenter les SGBD temps réel et leurs caractéristiques (cf. section II). Puis, nous présenterons les travaux que nous effectuons sur des modèles de SGBDTR basés sur l'utilisation d'une boucle de rétroaction pour contrôler le comportement des SGBDTR notamment durant les phases de surcharge et de sous-utilisation des systèmes (cf. section III). Nous conclurons cet article en donnant quelques pistes de recherche qui méritent d'être explorées dans les futurs travaux effectués sur les SGBDTR (cf. section IV).

II LES MODÈLES DE SGBD TEMPS RÉEL

Dans la littérature, on considère le plus souvent que les SGBDTR se trouvent en mémoire principale. Cela permet de s'abstraire de la problématique des entrées/sorties et des écritures différées sur disque. Un SGBDTR repose sur l'utilisation d'une base de données temps réel qui peut à la fois contenir des données classiques (dites non temps réel) et des données temps réel. Comme pour les SGBD classiques les unités de traitement sont les transactions mais elles peuvent

posséder des contraintes temporelles qui en font alors des transactions temps réel. Dans cette section, il s'agit ici de décrire les deux notions principales sur lesquelles reposent les SGBDTR : les données temps réel et les transactions temps réel.

II.1 Les données temps réel

La plupart des données manipulées par les applications temps réel sont issues de capteurs. Elles sont dites temps réel, c'est-à-dire qu'elles sont utiles au système seulement si elles sont utilisables et utilisées dans les temps. Chaque donnée temps réel possède une durée de validité. Les conséquences résultant de l'accès d'une transaction à une donnée en dehors de sa durée de validité dépendent des besoins particuliers de l'application et de la sémantique des données.

Un des nombreux problèmes soulevés par la conception de SGBDTR est le maintien de la cohérence des données dans la base [14, 15]. En effet, l'état de l'environnement tel qu'il est perçu par le système contrôleur doit refléter le plus fidèlement possible l'état réel de cet environnement. Cette exigence a une conséquence sur la conception de SGBDTR, qui doit non seulement respecter les contraintes d'intégrité (cohérence logique), mais aussi respecter les contraintes de cohérence temporelle des données. La cohérence temporelle regroupe en réalité deux notions [13] :

- la cohérence absolue qui est liée directement au fait que l'image de l'environnement dans le système doit refléter l'état réel de cet environnement,
- la cohérence relative qui est liée aux données utilisées pour dériver d'autres données dans la base qui doivent être cohérentes entre elles.

Pour illustrer ces deux notions, reprenons la définition d'une donnée temps réel d proposée dans [12] : $d = (d_{valeur}, d_{estampille}, d_{dva})$, où d_{valeur} est la valeur actuelle de la donnée d, $d_{estampille}$ représente l'instant où cette valeur a été mesurée ou calculée et d_{dva} est la durée de validité absolue de la valeur de d.

Un ensemble de cohérence relative R est l'ensemble des données utilisées pour dériver une nouvelle donnée. A tout ensemble R est associée une durée de validité relative R_{dvr}. Soit $d \in R$. On dit que d est dans un état correct, si et seulement si :

1. d_{valeur} est logiquement cohérente (elle satisfait aux contraintes d'intégrité),
2. d est temporellement cohérente :
   - cohérence absolue : (instant_courant - $d_{estampille}$) $\leq d_{dva}$
   - cohérence relative : pour tout $d' \in R$, $d_{estampille} - d'_{estampille} \leq R_{dvr}$

II.2 Les transactions temps réel

Une transaction temps réel effectue généralement une suite d'opérations de lecture ou d'écriture de données temps réel ou non temps réel en respectant une échéance (contrainte temporelle).

Selon les conséquences du manquement des échéances des transactions sur l'application et sur son environnement, les transactions dans un SGBDTR sont classées en trois catégories [6, 12] :

- Transactions à échéances strictes et critiques ("hard deadline transactions") : une transaction qui rate son échéance peut avoir des conséquences graves sur le système ou sur l'environnement contrôlé.
- Transactions à échéances strictes et non critiques ("firm deadline transactions") : si une transaction rate son échéance, elle devient inutile pour le système. Ses effets sont nuls. Elle est donc abandonnée dès qu'elle rate son échéance.
- Transactions à échéances non strictes ("soft deadline transactions") : si une transaction rate son échéance, le système ne l'abandonne pas immédiatement. En effet, elle peut avoir une certaine utilité pendant un certain temps encore après l'expiration de son échéance, mais la qualité de service qu'elle offre est moindre.

L'objectif global d'un SGBDTR est double : (1) la garantie absolue de l'exécution et de la validation (avant leurs échéances) des transactions à échéances strictes et critiques et (2) la minimisation du nombre de transactions à échéances strictes et non critiques ou à échéances non strictes qui ratent leurs échéances.

Les principaux travaux actuels sur les SGBDTR portent sur les systèmes où les

transactions possèdent des échéances strictes et non critiques ou des échéances non strictes. Ces travaux concernent particulièrement des techniques de contrôle de concurrence des transactions et leur ordonnancement ; l'objectif du système est de minimiser le nombre de transactions qui ratent leurs échéances. Pour les transactions à échéances strictes et critiques, il n'existe pas à notre connaissance d'algorithmes d'ordonnancement des transactions qui garantissent leur terminaison avant l'échéance. La principale raison provient de l'indéterminisme de ces transactions et du manque de méthodes permettant d'estimer avec précision le temps d'exécution des transactions.

Les transactions temps réel peuvent manipuler des données de base ou des données dérivées. Les données de base représentent des entités concrètes de l'environnement (température, pression, etc.). Les valeurs des données dérivés sont calculées à partir de celles d'autres données. Par exemple, les données acquises sur la position et la vitesse d'objets en mouvement sont utilisées pour dériver la nouvelle commande à appliquer au robot qui permet de déplacer ces objets. Cette distinction entre types de données conduit à un deuxième classement des transactions [1] :

- Transactions sensorielles : ce sont des transactions qui mettent à jour les données sensorielles. Elles sont à écriture seule.
- Transactions déclenchées de mise à jour : ce sont des transactions déclenchées par les mises à jour des données sensorielles. Elles mettent à jour les données dérivés. Ce sont donc des transactions en lecture/écriture.
- Transactions utilisateur : ce sont celles exécutées par l'utilisateur, avec des échéances. Elles sont en lecture/écriture.

III UN MODÈLE DE SGBDTR BASÉ SUR LA RÉTROACTION

Dans une application reposant sur l'utilisation de SGBDTR, des transactions en provenance des utilisateurs arrivent à des fréquences variables. Lorsque la fréquence augmente de façon considérable, l'équilibre du SGBDTR est mis en péril. Durant ces périodes de surcharge, le SGBDTR va potentiellement disposer de ressources moins importantes et les transactions temps réel vont alors manquer leur échéance en plus grand nombre. Des travaux basés sur une approche Qualité de Service (QdS) [8, 2] tentent de rendre les SGBDTR plus robustes face aux périodes d'instabilité (périodes de sous-utilisation et périodes de surcharge). Ces travaux s'appuient sur des techniques de contrôle avec rétroaction[1] [10] et autorisent la manipulation de résultats imprécis [9].

Dans cette section, nous allons détailler ces travaux sur lesquels s'appuient nos travaux [3, 7, 4] et plus particulièrement nous allons présenter le modèle de SGBDTR que nous considérons.

III.1 Les transactions temps réel et la qualité des transactions

Les transactions temps réel utilisées possèdent des échéances de type *firm* (à échéances strictes non critiques [6]). Par conséquent, lorsqu'elles dépassent leur échéance, elles deviennent inutiles pour le système et sont abandonnées. Dans notre cas, nous considérons deux types de transactions temps réel :

- Les transactions de mise à jour : elles ont pour tâche de mettre à jour régulièrement les données acquises auprès de capteurs. Ces transactions sont exécutées périodiquement pour rafraîchir la valeur des données temps réel.
- Les transactions « utilisateur » : elles effectuent des opérations de lecture/écriture de données non temps réel et/ou des lectures de données temps réel. La non prévisibilité de leurs arrivées dans le système et de la charge qu'elles suscitent, rend adéquate l'utilisation d'une architecture basée sur le retour d'expérience (ou rétroaction) pour gérer les variations importantes de charge [10].

La conception des transactions temps réel est basée sur des techniques de calculs imprécis [9]. Chaque transaction est composée d'une partie obligatoire et de plusieurs parties optionnelles qui sont exécutées suivant le temps qui est disponible pour l'exécution de la transaction. Plus le nombre de parties optionnelles exécutées avant échéance est grand, plus la qualité des transactions (QdT) est importante.

[1] Feedback Control Scheduling (FCS)

III.2 Les données temps réel et la qualité des données

Les données temps réel représentent la capture de l'état du monde réel. Par exemple, dans une centrale nucléaire, on peut trouver des capteurs de température qui sont utilisés pour contrôler le système et détecter les éventuelles anomalies (surchauffe du système ou autre). Ces données doivent être remises à jour régulièrement afin de refléter au plus près le monde réel. Elles possèdent donc une durée de validité qui représente la période pendant laquelle elles peuvent être utilisées.

Amirijoo et al. ont introduit une notion de la Qualité des Données (QdD) [2] qui permet de considérer qu'une donnée stockée dans la base peut posséder une certaine déviation par rapport à sa valeur dans le monde réel.

On appelle cette déviation "erreur sur la donnée" (notée DE[2]) et on la calcule en faisant la différence entre la donnée en base et la valeur du monde réel. Par exemple, on peut admettre que la donnée température lorsqu'elle vaut "37°2" en base est très proche de "37°3" qui est la donnée réelle et que, par conséquent, il n'est pas nécessaire de mettre à jour cette donnée.

Cette déviation possède un seuil (MDE[3]) qui permet de déterminer si la transaction qui souhaite mettre à jour une donnée temps réel peut être écartée (DE < MDE) ou pas (DE ≥ MDE). Ce travail est effectué par le *contrôleur de précision*.

III.3 Le modèle global

Dans cette section, nous allons présenter les différentes parties du modèle utilisées la gestion de la qualité de service basé sur une architecture de contrôle par rétroaction. La figure 1 donne une vision générale du modèle généralement utilisé dans ce type de travaux [2]. Nous allons donner une brève description de chacun des composants de ce modèle.

La tâche du *contrôleur d'admission* est de contrôler les transactions « utilisateur » qui sont acceptées ou pas dans le système. Il effectue ce contrôle en fonction de la charge d'utilisation calculée et des paramètres de qualité de service spécifiés par le DBA. Son fonctionnement est contrôlé par la boucle de rétroaction qui lui fournit ses paramètres de fonctionnement.

Les transactions qui sont admises dans le système sont placées dans une file d'attente avant d'être envoyées au *déclencheur de transactions*. Ce *déclencheur de transactions* possède pour fonction de gérer l'exécution des transactions. Il dispose de plusieurs modules complémentaires :

– Un *gestionnaire de fraîcheur* : il vérifie la fraîcheur des données qui vont être accédées par une transaction. Si les données sont obsolètes (non fraîches) alors la transaction est mise en attente dans une file.
– Un *contrôleur de concurrence* : il est chargé de gérer les conflits d'accès aux données qui apparaissent entre les transactions. Dans la plupart des travaux [8, 2], il s'agit du protocole 2PL-HP (*Two Phase Locking High Priority*) [1].
– Un *ordonnanceur de base* : il s'agit bien souvent de EDF (*Earliest Deadline First*) [5] qui ordonnance les transactions selon le principe que la transaction qui possède l'échéance la plus proche doit être exécutée en priorité.

Les différents modules du *déclencheur de transactions* peuvent être remplacés par des modules équivalents. C'est ainsi que l'on peut modifier la politique de contrôle de concurrence ou modifier la politique d'ordonnancement.

Un *moniteur* permet de mesurer les performances du système en inspectant l'exécution des transactions (quantité de transactions terminées, abandonnées, qui ont ratées leur échéance, ...). Les valeurs ainsi mesurées permettent d'alimenter le *contrôleur de qualité de service* et font parties de la boucle de contrôle par rétroaction qui va contribuer à stabiliser le système.

Le *contrôleur d'utilisation et d'échéances ratées* (ou *contrôleur de qualité de service*) permet de réajuster les paramètres de QdS en fonction des valeurs déterminées par le *moniteur* et des paramètres de référence[4] . Les valeurs ainsi obtenues sont transmises au *contrôleur d'admission* et au *gestionnaire de qualité des données*.

[2]Data Error
[3]Maximum Data Error
[4]fournit par le DBA

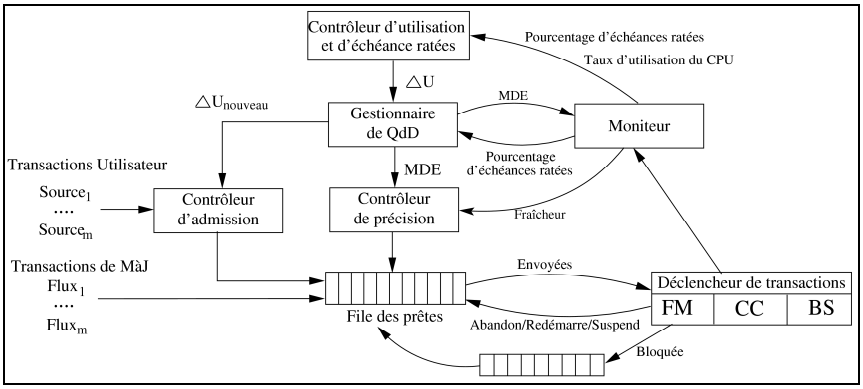

Fig. 1 : *Architecture de SGBDTR basée sur la rétroaction.*

Le *gestionnaire de qualité des données* permet de réajuster la valeur du paramètre MDE qui constitue le paramètre de QdD. La valeur de MDE est calculée en fonction de l'utilisation du système. Le paramètre MDE est ensuite fourni au *contrôleur de précision* qui écarte les transactions de mise à jour lorsque les données à mettre à jour sont suffisamment représentatives du monde réel en considération de la valeur de MDE.

III.4 La boucle de contrôle par rétroaction

La boucle de rétroaction possède pour tâche de stabiliser le système durant les phases d'instabilité. Pour cela, elle s'appuie sur le principe d'observation puis d'auto-adaptation. L'auto-adaptation a lieu tout au long du fonctionnement du système car les demandes des utilisateurs sont imprévisibles et la charge doit être ajustée en permanence. L'observation consiste à prendre en compte l'état de fonctionnement du système et à déterminer s'il correspond aux paramètres de qualité de service initialement spécifiés. Cette observation se fait via le *moniteur*. à partir de l'observation effectuée, le système adapte ses paramètres, via le *contrôleur de qualité de service*, afin d'augmenter ou de diminuer les transactions acceptées dans le système.

Le fonctionnement de la boucle de rétroaction doit tendre vers une stabilité du système autour d'une valeur de référence, fixée par le DBA (par exemple, il peut s'agir d'un taux d'utilisation de 80%).

III.5 Un contrôleur (m,k)-firm

Dans [7], nous avons surtout considéré les transactions de mise à jour pour contribuer à la stabilisation du système durant les phases de surcharge ou les phases de sous-utilisation. Notre contribution consiste surtout dans l'ajout d'un contrôleur (m,k)-firm pour les transactions de mise à jour.

III.6 Un modèle basé sur des données multi-versions

Dans [3, 4], nous avons présenté un modèle permettant de réduire le nombre de conflits d'accès et donc d'augmenter le nombre de transactions qui se terminent avant leur échéance. Ce modèle est basé sur la création de nouvelles versions des données lorsque se produit un conflit de type lecture/écriture ou écriture/lecture. Dans [3], nous avons utilisé un nombre fixe de versions pour l'ensemble des données alors que dans [4], le système est basé sur un ajustement dynamique du nombre de versions.

IV CONCLUSION ET PERSPECTIVES

Dans le domaine des SGBDTR, les travaux effectués concernent essentiellement les protocoles de contrôles de concurrence. La performance de ces protocoles est prouvée le plus souvent au moyen de simulations qui permettent de comparer de nouveaux protocoles par rapport à des protocoles de référence. Il s'agit en général d'améliorer les performances de ces protocoles en diminuant le nombre de transactions qui ratent leur échéance.

Des simulateurs permettent d'effectuer des études de performance des protocoles et des techniques développées pour les SGBDTR mais il n'existe pas encore de véritables SGBDTR ou de prototypes permettant de tester réellement toutes les techniques développées ces dernières années. Parmi les travaux de recherche qui devront être

envisagés, ceux concernant la conception de SGBDTR et d'applications reposant les SGBDTR apparaissent comme primordiaux pour faire avancer les recherches sur les SGBDTR.

D'autres travaux menés dans le domaine des SGBDTR permettent de prendre en considération d'autres particularités des applications [11]. Il s'agit par exemple des travaux menés sur les bases de données mobiles temps réel, des SGBDTR distribués ou répartis. Certains travaux concernent aussi des modèles de transactions tels que les transactions emboîtées temps réel.

L'étude des gestionnaires de recouvrement habituellement utilisés dans les SGBD classiques nécessitent une attention particulière dans les SGBDTR. En effet, dans les SGBD classiques ils permettent au moyen de fichiers journaux d'effectuer des reprises après panne. Dans les SGBDTR, de par la nature temps réel des transactions, certaines ne nécessitent par d'être récupérées (ré-exécutées) lors de la reprise après panne.

V REMERCIEMENTS

Ces travaux ont été effectués grâce à la contribution financière du Ministère Français de la recherche (ACI-JC 1055).

BIBLIOGRAPHIE

[1] R. Abbott and H. Garcia-Molina. Scheduling Real-Time Transactions : A Performance Evaluation. In *Proc. of the 14th Intl. Conf. on Very Large Data Bases (VLDB)*, pages 1–12, Los Angeles, California, 1988.

[2] M. Amirijoo, J. Hansson, and S.H. Song. Algorithms for Managing QoS for Real-Time Data Services Using Imprecise Computation. In *Proc. of Intl. Conf. on Real-Time and Embedded Computing Systems and Applications (RTCSA'03)*, Taiwan, R.O.C., 2003.

[3] E. Bouazizi, C. Duvallet, and B. Sadeg. Utilisation de données multi-versions et de l'ordonnancement contrôlé par rétro-action pour les transactions temps réel. In *Actes de conférence de MajecStic'2004 (MAnifestation des JEunes Chercheurs STIC)*, page sur cdrom, Calais, France, 13 au 15 octobre 2004.

[4] E. Bouazizi, B. Sadeg, and C. Duvallet. Ordonnancement contrôlé par rétroaction dans les sgbd temps réel. In *Actes de conférence de RTS'200S (Real-Time and Embedded Systems)*, à paraître, Paris, France, 5-6 avril 2005.

[5] G.C. Buttazzo. *Hard Real-Time Computing Systems*. Kluwer Academic Publishers, 1997.

[6] C. Duvallet, Z. Mammeri, and B. Sadeg. Les SGBD temps réel. *Technique et Science Informatiques (TSI)*, 18(5) : 479–517, 1999.

[7] J. Haubert, E. Bouazizi, and C. Duvallet. Utilisation de contraintes (m,k)-firm dans les sgbdtr manipulant des transactions périodiques de mise-a-jour. In *Actes de conférence de MajecStic'2004 (MAnifestation des JEunes Chercheurs STIC)*, page sur cdrom, Calais, France, 13 au 15 octobre 2004.

[8] K. Kang, S.H. Son, J.A. Stankovic, and T.F. Abdelzaher. A QoS-Sensitive Approach For Timeliness and Freshness Guarantees in Real-Time Databases. In *Proceedings of the Euromicro Conference on Real-Time System*, 2002.

[9] J.W.S. Liu, W.-K. Shih, and J.-Y. Chung K.-J. Lin, R. Bettati. Imprecise Computations. In *Proceedings of the IEEE*, volume 82, pages 83–94, january 1994.

[10] C. Lu. *Feedback Control Real-Time Scheduling*. PhD thesis, University of Virginia, 2001.

[11] K. Ramamritham, S.H. Son, and L.C. DiPippo. Real-Time Databases and Data Services. *Real-Time Systems*, (28) : 179–215, 2004.

[12] Krithi Ramamritham. Real-time databases. *Distributed and Parallel Databases (Invited Paper)*, 1(2) : 199–226, 1993.

[13] S. H. Son, J. Lee, and Y. Lin. Hybrid protocols using dynamic adjustement of serialization order for real-time concurrency control. *Journal of RTS*, 4(3) : 269–276, 1992.

[14] S.H. Son and J.W.S. Liu. Maintaing temporal consistency : pessimistic vs. optimistic concurrency control. *IEEE Transactions on Knowledge and Data Engineering*, 7(5), 1995.

[15] M. Xiong, J.A. Stankovic, K. Ramamritham, D. Towsley, and R.M. Sivasankaran. Maintaining temporal consistency : issues and algorithms. In *First Int. Workshop on RTDBS. Issues and*