システムソフトウェアとオペレーティング・システム 86−13
(2001. 3. 6)

リアルタイムデータベースのための データ品質管理手法の提案

川島 英之† 遠山 元道‡ 安西 祐一郎‡

†慶應義塾大学大学院 理工学研究科 計算機科学専攻
‡慶應義塾大学 理工学部 情報工学科
〒223-8522 横浜市港北区日吉 3-14-1
E-mail: {kawasima@ayu,toyama@db,anzai@ayu}ics.keio.ac.jp

あらまし 本稿では、我々の提案手法 **Multi-level Virtual Deadline of Updates(MVDU)** の 2 つの拡張について述べる。まず、**MVDU** の複数センサ複数クライアントへの適用方針を示す。アップデート制御実験にもとづく我々の方針は、いずれのクライアントにも temporal consistency 違反を起こさせず、また real-time 制約違反に敏感に反応するものである。次に、頻繁に到着するアップデートをストレージにバルク転送する機構 *lazy synchronization* を提案する。従来手法との比較実験により提案機構の優位性が示される。

キーワード リアルタイムデータベース ヴァーチャルデッドライン レイジィシンクロナイゼイション

A Proposal of a Method to Manage Data Quality for Real-Time Databases

Hideyuki Kawashima† Motomichi Toyama‡ Yuichiro Anzai‡

†Department of Computer Science, Faculty of Science and Technology, Keio University
‡Department of Information and Computer Science, Faculty of Science and Technology, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522 Japan

Abstract In this paper, we describe 2 extensions of **Multi-level Virtual Deadline of Updates(MVDU)** which we proposed before. First, we show the policy of **MVDU** for multi-sensors and multi-clients. Our policy prevents temporal consistency constraint violations and quickly responses to real-time constraint violations. Second, we propose the *lazy synchronization* architecture which collectively transmits sensor updates into storage. And we compare the architecture with conventional one. The experimental result shows the superiority of our proposal.

Keywords real-time databases virtual deadline lazy synchronization

1 はじめに

リアルタイムデータベースはリアルタイムシステムとデータベースシステムの混合システムである。その開発例には、商用例としてClustRa[1]、EagleSpeed[2]等があり、研究例にはSTRIP[3]、StarBase[4]、BeeHive[5]等があるが、その数はまだ少ない。

開発例が少ない理由の1つに、リアルタイム環境はオペレーティングシステムさえオーバヘッドだと考えられていることが挙げられる[6]。

しかし、そのような考え方はアーキテクチャの進歩を無視してはいないだろうか。将来、高性能なリアルタイム支援アーキテクチャが開発されたとき、通常の計算システム同様に、必ずデータベースは求められることだろう。現在はCPUの性能向上、不揮発性RAMの開発に加えて、リアルタイムアプリケーション[7]を支援する基盤アーキテクチャ[8]も開発され始めている。我々は、リアルタイムデータベースが必要とされる日は近いと考えている。

2 関連研究および本研究の寄与

現在のリアルタイムデータベース研究動向に目を転じると、その着眼点のほとんどはクライアントアクセスの制御である。すなわちリアルタイム並行実行制御[9][10]、リアルタイムトランザクション処理[11][12]、そしてインプリサイス計算[13][14]等である。

しかし、STRIP[3]のように開放環境に置かれるとき、またcontinual query[15]のように時々刻々と変化する動的環境を監視するとき、リアルタイムデータベースは頻繁に到着するセンサアップデートをも扱わなければならない。これは更新処理への計算資源割り当てを迫り、クライアントが発行したトランザクションへ割り当てる計算資源を減少させる。従ってこのような環境下ではリアルタイムデータベースはクライアントからのトランザクションのみならず、センサアップデートも制御すべきである。

センサアップデートの間隔とクライアントが求めるtemporal consistency制約の間に生じるギャップを埋めることで、計算資源をクライアントからのトランザクションに割り当てることができる。

この観点のもと、Dattaらは97年にtemporal consistency制約が守られる範囲でセンサアップデート許可量を制御することでreal-time制約を満たす先駆的手法**virtual deadline of updates(VDU)**を提案した[16]。しかしながら、我々はこの手法に次の欠点を見出した。

まず、センサアップデートを控えるパラメータ*virtual deadline*を安定させる考えが欠落していること。この欠点はtemporal consistency制約違反を招く。次に、彼らのモデルは周期的センサアップデートに対しては不適切であり、計算資源の無駄遣いを招くこと。最後に、彼らは提案手法の評価をシミュレーション実験でしか行なっていないこと。プラットフォームの変化はシステムの挙動に大きな影響を及ぼすゆえ、我々は**VDU**のように繊細な手法をシミュレーションにより評価することは難しいと考える。

これらの欠点に対し、我々は**VDU**を周期的アップデート環境に適応させる手法**Multi-level Virtual Deadline of Updates(MVDU)**[17]を提案した。

本研究では**MVDU**を拡張し、次の寄与をおこなう。

1. **MVDU**を複数センサ複数クライアントへの対応させる手法の提案
2. 低コストにより時間的データ品質劣化を防ぐ機構*lazy synchronization*の提案

3 リアルタイムデータベースが守るべき制約

3.1 Real-time制約

Real-time制約が満足される事と応答時間が短い事とは同意義ではない[18]。RTDBがデッドライン以内に応答するとき、RTDBにおいてreal-time制約が満たされるといわれる。real-time制約にはhard-critical、hard-essential、firm、softの4種類がある[19]。このうち本研究ではhard-critical(図1)以外のreal-time制約を対象とする。hard-critical real-time制約におけるモデルでは高々一度のデッドラインミスによりシステムに破滅的な損害がもたらされる。

以降ではhard-critical real-time制約以外のreal-time制約をRT制約とする。

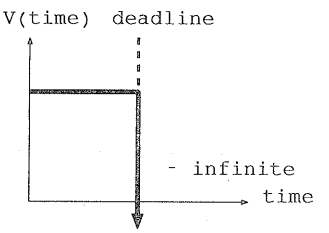

図1: Hard-critical real-time制約

3.2 Temporal consistency制約

DBから供給されたデータの発生時刻が要求時刻から一定時間以内であるとき、reasonable absolute temporal consistency制約が満たされるといわれる。

Temporal consistency制約には、要求時刻からの近さ(absolute)と返却されるデータ群の同期度(relative)の2種類がある[19][16][20]。本研究ではまず、データ項目が1つの場合のみを対象とするために、要求時刻からの近さ(absolute)のみを対象とする。

そしてRT制約とreasonable absolute temporal consistency制約の両方を満たすために、perfect absolute temporal consistency制約を満たすことは諦め、reasonable absolute temporal consistency制約を満たすことを目標とする。

さて、Dattaらの研究[16]に基づきreasonable absolute temporal consistency制約を次のように定義する。

定義: Reasonable absolute temporal consistency制約

- D_i^v がその発生時に押されたタイムスタンプを $DTS(D_i^v)$ とする。
- データ項目 i の版 v を D_i^v とする。
- そしてクライアントにより許容される発生時刻と要求時刻の最大のずれを *Maximum Tolerable Gap(MTG)* とする。

このとき次式が成り立つときに限り D_i^v は t_j についてreasonable absolute temporal consistency制約を満たすとする。

$$|DTS(D_i^v) - t_j| \leq MTG$$

以降ではreasonable absolute temporal consistency制約をTC制約とする。

3.3 両制約を同時に満足させる手法

センサが頻繁に更新される環境において、センサアップデートとクライアントトランザクションの優先度を制御する手法として、**Do Updates First(UF)** と **Do Transactions First(TF)** がSTRIP研究グループにより提案された[21]。

しかし、これらの手法はTC制約とRT制約を同時に満足させることを考慮していなかった。これに対してDattaらの提案手法 **VDU** はアップデート制御により両制約を同時に満足させる。

VDU はアップデートを *virtual deadline* と呼ばれる時間だけ拒絶することにより、データ品質を制御する。TC制約が満足されない時には *virtual deadline* を引き下げアップデートが許可される割合を増やし、逆にRT制約が満足されない時には *virtual deadline* を引き上げ、アップデートが許可される割合を減らして、トランザクションに資源が回るようにする。

VDU は非周期的アップデートのために提案された手法であった。これに対し、我々は限定された環境である、周期的アップデートに対して有効な手法 **MVDU**[17] を提案した。

4 Multi-level Virtual Deadline of Updates(MVDU)

4.1 手法

MVDU ではアップデートを許可する割合 *virtual deadline* を離散化し、*level of VD* とする(図2)。**VDU** 同様に、アップデートは *level of VD* 以降ならば許可されるが、*level of VD* 以前ならば拒絶される。

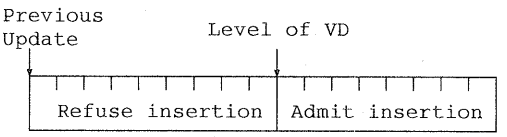

図 2: Level of VD

我々は *level of VD* の間隔を次のように設定した。

Level of VD の差 = アップデート周期

我々の想定環境は周期的アップデートであるために、*level of VD* の差は等しく一定になることを指摘しておく。そしてRT制約を優先するか、TC制約を優先するかにより異なる手法をとる。以降の小節においてそれらを述べる。

4.1.1 RT制約を優先する場合

両制約を満たす範囲でRT制約を最も満足する手法を図3に示す。

```
Level of VD を 2 * MTG に設定する;
データ読み出しごとに次の処理を行なう;
if (RT制約が満足)
  if (TC制約が不満足)
    Level of VD をひとつ下げる;
  else
    なにもしない;  /* 両制約共に満足 */
else
  Level of VD をひとつ上げる;
```

図 3: **MVDU** (RT制約優先)

level of VD の初期値は *MTG* の2倍に設定される。この *level of VD* はTC制約がようやく満たされる値である。なぜならば、要求時刻以前に存在する最近傍データの発生時刻も要求時刻以後に存在する最近傍データ発生時刻も *MTG* 以内であればTC制約は満たされるからである。

そしてRT制約が満たされ、かつTC制約が満たされない場合には *level of VD* はひとつ下げられる。こ

うして *level of VD* は、両制約が満たされる中で最もRT制約が満たされる値に収束する。RT制約が満たされない場合は、TC制約が満たされていようといまいと *level of VD* はひとつ上げられる。

4.1.2 TC制約を優先する場合

両制約を満たす範囲でTC制約を最も満足する手法を図4に示す。

```
Level of VD を最小値 (0) に設定する;
データ読み出しごとに次の処理を行なう;
if (TC制約が満足)
  if (RT制約が不満足)
    Level of VD をひとつ上げる;
  else
    なにもしない;  /* 両制約共に満足 */
else
  Level of VD をひとつ下げる;
```

図 4: **MVDU** (TC制約優先)

最初に *level of VD* の初期値を最小に設定する。このときTC制約は最もよく満たされ、RT制約は最も満たされにくい。

そしてTC制約が満たされ、かつRT制約が満たされないならば、*level of VD* をひとつ上げてRT制約を満たされやすくする。こうして *level of VD* は、両制約が満たされる中で最もTC制約が満たされる値に収束する。ただしTC制約が満たされないならば、RT制約が満たされていようといまいと *level of VD* はひとつ下げられる。

section 複数センサ複数クライアントへの適用センサ及びクライアントが複数存在する環境に **MVDU** を適用させるとき、次の2つの問題が発生する。

1. あるセンサ系列に対してクライアントが複数存在するとき、最大でクライアント数だけの MTG が存在する可能性がある。しかし、そのセンサ系列が持つことができる *virtual deadline* は高々1つである。そのセンサ系列の *virtual deadline* はどのようにして決められるべきか？
2. あるクライアントが複数のセンサ系列にアクセスする時、要求時刻とセンサデータ発生時刻との近さだけでなく、複数センサ系列の同期もまたクライアントにより要求される。この要求を満たすためには、各センサ系列の *virtual deadline* はどのように取り扱われなければならないか？

これらの問題への我々の対策を以下の小節において述べる。その準備のために、MTG を2つの概念に分解し、明確性を与える。まず、我々が今までTC制約と呼称してきた制約を、resonable absolute temporal consistency制約とし、これ以降ATC制約とする。

次に、resonable relative temporal consistency制約をDattaらの研究 [16] に基づき次のように定義する。

定義: Reasonable relative temporal consistency 制約

- D_i の版 v を D_i^v とする。
- D_j の版 w を D_j^w とする。
- D_i^v がその発生時に押されたタイムスタンプを $DTS(D_i^v)$ とする。
- D_j^w がその発生時に押されたタイムスタンプを $DTS(D_j^w)$ とする。
- クライアントにより許容されるセンサデータ発生時刻の最大のずれを *relative maximum tolerable gap(RMTG)* とする。

このとき次式が成り立つときに限り D_i^v と D_j^w は *reasonable relative temporal consistency* 制約を満たすとする。

$$|DTS(D_i^v) - DTS(D_j^w)| \leq RMTG$$

これ以降、reasonable relative temporal consistency制約をRTC制約と呼ぶ。

4.2 複数クライアントへの対応

さて、1番目の問題に対して、我々はセンサ系列 D_i が持ち得る最大の *virtual deadline* を次のように設定することで解決を図る。

定義: *Max (virtual deadline*(D_i)*)*

- センサ系列 D_i の *virtual deadline* を *virtual deadline*(D_i) とする。
- D_i にアクセスするクライアント C_j の $AMTG$ を $AMTG(D_i \leftarrow C_j)$ とする。

このとき $Max(virtual\ deadline(D_i))$ を次のように定める。

$$Max(virtual\ deadline(D_i)) = min(AMTG(D_i \leftarrow C_j)))$$

すなわち、D_i にアクセスするクライアントがもつ $AMTG$ の中で最小の値が $max(virtual\ deadline(D_i))$ として選択される。$AMTG$ が小さいクライアントほどATC制約は厳しい。そこで、D_i にアクセスする全てのクライアントにATC制約違反に起こさせないために、我々は最も厳しい値を $max(virtual\ deadline(D_i))$ に設定する。この結果 $max(virtual\ deadline(D_i))$ が比較的小さくなったならば、アップデートを制限することでRT

制約を満たさせるという **MVDU** の目的が十分果たされなくなる可能性がある。このような場合には、トランザクション制御に頼るほかないと我々は考えている。

4.3 どのようにして relative temporal consistency 制約を満たすか？

非周期的アップデート環境においてRTC制約が満足されているか調べるためには、全センサ系列に少なくとも一度はアクセスしなければならない。センサ系列が n 本あるとき、この計算量は $O(n)$ となる。

しかし、周期的アップデート環境において最新のデータのみを監視するならば、*AMTG* と *RMTG* に次のような制約を与えることで、アップデートに故障がおきない限り、RTC制約は必ず満たされる。

1. *AMTG* は、そのセンサ系列の周期以上でなければならない。
2. *RMTG* は、トランザクションがアクセスするセンサ系列の周期の中で、最大の周期以上でなければならない。

1つ目の制約が満足されなければ、トランザクション処理にあたり、必ず *AMTG* 違反が発生する。2つ目の制約が満足されなければ、トランザクション処理にあたり、必ず *RMTG* 違反が発生する。

このときRTC制約は必ず満足されるから、それが満たされているか調べる処理は不要になる。従って、

トランザクション処理毎に $O(n)$ の計算量が非周期的アップデートの場合に比べて不要になる。

4.4 複数センサ環境における *virtual deadline* の変更方針

複数センサ環境において、どのように *virtual deadline* を変化させることがシステムにとって有効であるのか？多くのセンサ系列の *virtual deadline* を変化させることは、システムの挙動に大きな影響を与える。

高々一度のミスにより、すべてのセンサ系列の *virtual deadline* を変化させるべきであるのか？ それとも次のようなパラメータを考慮すべきなのか？

- ミスしたtemporal consistencyの種別

  2つのabsoluteとrelativeによりミスの度合を変化させるべきか？ ATC制約は高々1つのセンサ系列にしか関係しないのに対して、RTC制約は少なくとも2つのセンサ系列に関係する。RTC制約が満たされない場合の *virtual deadline* の変化量は、ATC制約が満たされないときのそれに比べて大きいべきだろうか？

- アクセスしたセンサ系列の数

  多数のデータにアクセスしたトランザクションは、システムの挙動を変化させるにあたり、大きな影響をもつべきか？ また、トランザクションがアクセスした全てのセンサ系列の *virtual deadline* を変化させるべきだろうか？

- 優先度

  トランザクションに与えられた優先度により、変化させるデータ系列の本数を、また *virtual deadline* を変更させる度合を変更するべきか？

これらのパラメータを考慮することは重要である。しかし、本論文では *virtual deadline* の変更方針について次のように言及するにとどめる。

複数センサ環境における
***virtual deadline* の変更方針**

トランザクションがATC制約もしくはRTC制約を満たせなかった場合には、すべてのセンサ系列の *virtual deadline* を同方針に基づき増減させる。

我々がこのように方針を定める理由は、アップデート量を減らすことによりRT制約が著しく満たされやすくなるわけではないことを示す実験結果が存在することである。

我々はアップデート拒絶量がRT制約に与える影響を調べるために、ATC制約を優先した場合とRT制約を優先した場合とで、応答時間がどのように変化するかを、表1に示す4条件下で調査した。

表 1: *AMTG* と *deadline* の設定

条件	*AMTG*	*Deadline*	条件の意図
A	10 s	10 s	両制約共に緩
B	10 s	0 s	TC 緩、RT 厳
C	1 ms	10 s	TC 厳、RT 緩
D	1 ms	0 s	両制約共に厳

実験に用いたシステムの構成を図5に示す。

本実験におけるアップデート及びトランザクションの周期、そして問い合わせ内容を表2に示す。

ここでは紙面の都合から条件Aの結果のみ記載するが、他の結果もほぼ同様である。図6において、上図はATC制約を優先した場合の結果を表し、下図はRT制約を優先した場合の結果を表している。

ATC制約を優先した場合には *virtual deadline* が比較的低く設定される。この時にはアボートされるアップデートの数が少なくなり、ATC制約が満たされやすくなる一方、RT制約が満たされにくくなる。

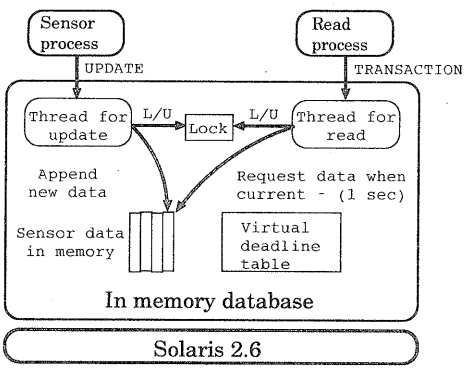

図 5: アップデート拒絶と応答時間の関係を調べるための実験システム構成図

表 2: 周期及び問い合わせ内容

クライアント数	1
センサ数	1
アップデート周期	1 ms
トランザクション周期	1 ms
実験回数	32768
問い合わせ内容	「1 秒前のデータを供給せよ」

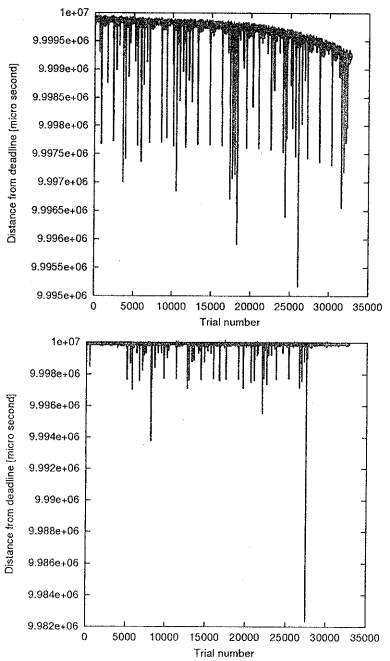

図 6: 条件 A における応答時間の遷移 (上:TC 制約優先、下:RT 制約優先)

RT 制約を優先した場合には *virtual deadline* が比較的高く設定される。この時にはアボートされるアップデートの数が多くなり、TC 制約が満たされにくくなる一方、RT 制約が満たされやすくなる。

これらの実験結果は次の事実を示している。

1. アップデート許可量を減らすことにより、最良実行時間の劣化を鈍らすことができる。
2. アップデート許可量を減らすことにより、悪いパフォーマンスが発生する割合が減る。
3. アップデート許可量を減らしても、最悪実行時間はほとんど改善されない。逆に悪くなる場合もある。

これより、我々はアップデート許可量を大幅に減らすことによって、ようやく RT 制約違反数を減らすことができるとの見解に達した。センサ系列が異なるポリシで動いていては、アップデートを大幅に変更することは困難である。

従って、複数センサが存在する環境においては、全センサ系列が同ポリシを持って統一的に *virtual deadline* を変化させなければ、RT 制約違反時にそれを改善することは難しいと考える。

5 Lazy synchronization

ディスクへの書き込みは次の 2 つの性質をもつ。

1. 一度にアクセスする時間が長い。
2. アクセス時間は転送量に比例しない。

まず、この性質を示す実験結果を表 3に示す。

表 3: ディスク性能実験

1 度の転送量	転送回数	総転送量	所用時間
1 バイト	1M 回	1M バイト	約 9300 ms
1M バイト	1 回	1M バイト	約 10 ms
1 バイト	1 回	1 バイト	約 0.1 ms

この実験から次のことがわかる。

1. 1M バイトを転送するにあたり、バルク転送は細切れ転送よりも 900 倍以上高速である。
2. 1 アクセスによる所用時間は、転送量に 1M 倍の差があっても 100 倍にしかならない。

これより、我々はアップデートをストレージに一括して転送する機構 *lazy synchronization* を考案した。**VDU** では *virtual deadline* 以前にシステムに到着したアップデートはアボートするが、それは将来に発生する過去データ探索においてデータ品質の劣化を招いてしまう。提案機構の目的はわずかなオーバヘッドによりその劣化を防ぐことである。

この提案機構におけるアップデートの処理手順を次に述べる。この処理はアップデート到着時刻が *virtual deadline* 以前であるか以後であるかにより異なる。

Virtual deadline 以前に到着したアップデート

1. アップデートは abort buffer の末尾に追加される。

Virtual deadline を越えて到着したアップデート

1. まず、アップデートは abort buffer の末尾に追加される。
2. abort buffer 中の全データがストレージに一括転送される。
3. abort buffer 中のデータがクリアされる。
4. cache buffer の末尾にアップデートが追加される。

5.1 比較実験

我々は *Lazy synchronization* を図 7に示す機構において実験した。この機構では、両制約が満たされながら、TC 制約が優先される。

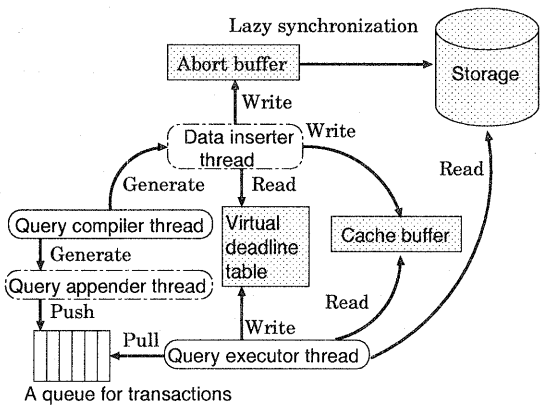

図 7: *Lazy synchronization* 評価用機構

そして比較実験をおこなうために、**UF** を図 8のように実装し、アップデートに要する時間を比較した。

アップデート周期が100μs、1ms、10ms、そして100msである際に、1024 回のアップデートを終了させるのに要した時間を表 4に示す。アボートパラメータ *virtual deadline* はいずれの場合も 1 秒に設定した。表 4より、*lazy synchronization* を伴う際の処理速度は **UF** のそれを明らかに上回ることがわかる。そして、*lazy synchronization* がある場合とない場合の処理時間を比較すると、オーバヘッドが極めて少ないこともまた示される。

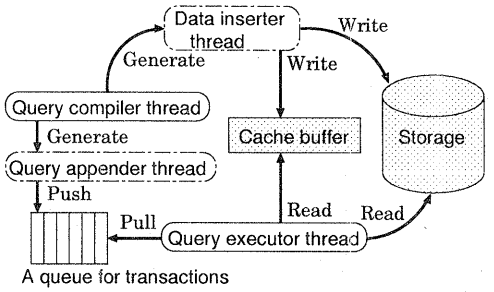

図 8: **UF** 評価用機構

表 4: *Lazy syncronization* の評価

評価手法	100 μs	1 ms	10 ms	100 ms
UF	14 s	25 s	31 s	123 s
提案機構有	5 s	20 s	25 s	112 s
提案機構無	3 s	20 s	21 s	112 s
待機時間	約 0.1 s	約 1 s	約 10 s	102.4 s

6 課題

- *Virtual deadline* の変更方針

  Virtual deadline の詳細な変更方法について、本稿ではおおまかな方針を提案することしかできなかった。我々は第 4節において、*virtual deadline* 変更にあたり考慮すべき 3 つのパラメータについて述べたが、これについてより考察を深めなければならない。

- *Lazy synchronization* と durability の関係

  Lazy synchronization はわずかなコストを支払うことでデータ品質の劣化を防ぐことができることを示したが、abort buffer に入れられたセンサデータは、システム異常時には消去されてしまう。それゆえ ACID 特性のひとつ durability が保てない。これへの対策として、我々は不揮発性 RAM を用いる必要があるのではないかと考えている。

- 周期と非周期の関係

  我々の対象環境は周期的アップデートである。これは非周期的アップデートの 1 ケースに過ぎない。しかしながら、この環境においてはさまざまな利点がある。本研究では temporal consistency 違反に関する利点を述べた。今後もこの利点を明確化していく。

- アップデート監視問い合わせ機構の導入

  アップデートを外部プロセスが監視する際にはプロセス間通信が余計なオーバヘッドとなる。Liu らはデータベースの状態を継続監視する機構 continual query[15] を提案している。今後、オーバヘッドの低い監視機構として、周期的アップデート環境における、アップデート監視クエリ機構を考えていきたい。

7 まとめ

本稿では、まず **MVDU** を複数センサ複数クライアントへ拡張する方針を示した。次に、時間的データ品質劣化を低オーバヘッドで防ぐ機構 *lazy synchronization* を提案した。比較実験により、提案機構の有効性を示すことができた。しかし、今回我々が述べた複数センサ複数クライアントの拡張はまだ不十分である。今後、さらに掘下げた研究をおこなっていく。また、周期的アップデートは非周期的アップデートに比較して遥かに扱い易い環境である。この環境に対して一層の研究をおこない、非周期に対する優位性を明確化していきたい。

参考文献

[1] Svein-Olaf Hvasshovd, Oystein Torbjornsen, Svein Erik Bratsberg, and Per Holagerf. The ClustRa Telecom Database: High Availability, High Throughput, and Real-Time Response. In *Proceedings of the 21st VLDB Conference*, pp. 469–477, Zurich, Switzerland, 1995.

[2] http://www.lmco.com/orss/eaglespeed, July 1998.

[3] B. Adelberg, H. Garcia-Molina, and B. Kao. Emulating soft real-time scheduling using traditional operating system schedulers. In *IEEE Real-Time Systems Symposium*, 1994.

[4] A. Bestavros, K. J. Lin, and S. H. Son, editors. *Developing a Real-Time Database: The StarBase Experience*. Kluwer Academic Publishers, 1997.

[5] J. Stankovic and S. H. Son. Architecture and Object Model for Distributed Object-Oriented Real-Time Databases. In *Proceedings of IEEE Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'98)*, pp. 414–424, Kyoto, Japan, April 1998.

[6] 徳田英幸, 加藤和彦, 越塚登, 石川裕, 岡村英明. ポスト PC 時代における OS 研究開発の重要性. 情報処理, Vol. 41, No. 10, pp. 1182–1187, October 2000.

[7] Y. Anzai. Human-Robot-Computer Interaction: A New Paradigm of Research in Robotics. *Advanced Robotics*, Vol. 8, No. 4, pp. 357–369, August 1994.

[8] N. Yamasaki. A functionally distributed responsive micro controller for distributed real-time processing. In *Proceedings of IEEE/RSJ International Conference on Inteligent Robots and Systems*, pp. 793–798, 1997.

[9] Robert Abbott and Hector Garcia-Molina. Scheduling Real-Time Transactions: a Performance Evaluation. In *Proceedings of the 14th VLDB Conference*, pp. 1–12, Los Angels, California, April 1988.

[10] Juhnyoung Lee and Sang H. Son. Performance of Concurrency Control Algorithms for Real-Time Database Systems, April 1994.

[11] J. Haritsa, M. Livny, and M. Carey. Earliest deadline scheduling for real-time database systems, 1991.

[12] A. Datta, S. Mukherjee, P. Konana, I. Viguier, and A. Bajaj. Multiclass transaction scheduling and overload management in firm real-time database systems. *Information Systems*, Vol. 21, No. 1, pp. 29–514, 1996.

[13] Kwei-Jay Lin, Swami Natarajan, and Jane W.-S.Liu. Concord: A System of Imprecise Computation. In *Proceedings of the Eleventh Annual International Computer Software and Applications Conference (COMPSAC 87)*, pp. 75–81, 1987.

[14] Susan V. Vrbsky. APPROXIMATE-A Query Processor That Produces Monotonically Improving Approximate Answers. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, pp. 1056–1068, December 1993.

[15] Ling Liu, Calton Pu, and Wei Tang. Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 124, pp. 139–152, 1999.

[16] Anindya Datta and Igor R. Viguier. Providing Real-Time Responce, State Recency and Temporal Consistency in Databases for Rapidly Changing Environments. *Information Systems*, Vol. 22, No. 4, pp. 171–198, 1997.

[17] 川島英之, 遠山元道, 安西祐一郎. Multi-level Virtual Deadline: Temporal Consistency と Real-Time を同時に満足する資源割り当て手法. In *Proceedings of the Advanced Database Symposimum*, December 2000.

[18] J. Stankovic, S. Son, and J. Hansson. Misconceptions About Real-Time Databases. *IEEE Computer*, Vol. 32, No. 6, pp. 29–36, June 1999.

[19] Joakim Eriksson. Real-Time and Active Databases: A Survey. In *Lecture Notes in Computer Science 1553: Active, Real-Time and Temporal Database System*, pp. 1–23, September 1997.

[20] 宗像浩一, 吉川正俊, 植村俊亮. 鮮度と同期度に基づく周期データの選択方式. 情報処理学会論文誌 データベース, Vol. 41, No. SIG1 (TOD5), pp. 140–153, February 2000.

[21] B. Adelberg, H. Garcia-Molina, and B. Kao. Applying update streams in a soft real-time database system. In *ACM SIGMOD Proceedings*, 1995.