[C1-5] メモリロギングによるセンサデータ挿入処理の高速化

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
開放環境科学専攻 コンピュータ科学専修
〒 223-8522 神奈川県横浜市港北区日吉 3-14-1

概要

頻繁に新しいデータが到着するようなセンサデータベースシステムでは、クライアントにとってデータ鮮度が高い方が望ましい。このため、センサデータベースへの挿入に要する時間は短いほど望ましい。そこで本研究では、従来のデータ挿入処理のボトルネックであるディスクロギングに注目し、1 台のリモートノードのメモリにネットワークを介してログを書くシングルメモリロギングを評価する。そしてシングルメモリロギングを高速化し、メモリログの永続性を強化するマルチメモリロギングを提案する。高速化には (1)UDP を用いる手法と、(2)BPF(Berkeley Packet Filter) を経由する手法のふたつのアプローチを試す。永続性を強化するために、3 台のログサーバとそれらを管理するログサーバマネージャを導入する。ディスクロギングに比べて、シングルメモリロギングは最大で約 5.20 倍、マルチメモリロギングは最大で約 1.67 倍速いことを実験により示す。

キーワード　メモリロギング　データ鮮度　リアルタイムデータベースシステム

Fast sensor data insertion using memory logging

Hideyuki KAWASHIMA　Motomichi TOYAMA　Yuichiro ANZAI

Computer Science, School of Science for Open and Environmental Systems, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522 Japan

Abstract

The sensor database system to which new sensor data frequently arrive should provide fresh data to its clients. Toward this end, fast sensor data insertion to the database is essential. The bottleneck of the conventional sensor data insertion is disk logging that writes a log record to disk before accessing buffer pool in memory. The purpose of this paper is to show fast sensor data insertion using memory logging that writes a log record to neighboring nodes' memory. We further propose multiple memory logging which improves the speed of memory logging and the persistence of the memory log records. To accomplish speed improvement, we try two approaches, (1) UDP(User Datagram Protocol), and (2) BPF(Berkeley Packet Filter). To accomplish persistence improvement, we introduce several log servers and a log server manager that manages them. The result of our experiment shows that single memory logging is about 5.20 time and multiple memory logging is about 1.67 times faster than disk logging in the maximum case.

Keywords　memory logging　freshness of data　real-time database system

1. はじめに

センサデータを実時間処理するシステムが現れつつある。たとえば、株価を実時間分析するシステムは数多く存在する。また、多くのヒューマンコンピュータインタラクションシステムには実時間性が求められる [1]。ノードブラウザ [2] は複数の移動センサノードから得たセンサ情報を実時間統合し、ブラウザへ表示する。

これらのシステムは複数のセンサ情報をもとに処理を行うために、データベースを使う必要がある。そしてこれらのシステムは、データベースシステムに対して、通常機能を提供することに加えて、実時間性制約と時間一貫性制約を保証することを要求する [3]。データベースシステムが実時間性制約を保証するには、クライアントにデータを提供する時刻を、クライアントが定める期限であるデッドライン以前にしなければならない。データベースシステムが時間一貫性制約を保証するには、クライアントに提供するデータまたはデータ集合の発生時刻とクライアントの求める時刻の差を、クライアントが許容するずれ以内に収めなければならない。時間一貫性には、ひとつのデータの発生時刻とクライアントの要求時刻とのずれである絶対時間一貫性と、データ集合の同期度である相対的時間一貫性がある。絶対時間一貫性のなかに、 データ発生時刻と現時刻との差である、データ鮮度がある。

データ鮮度が低ければ、ノードブラウザは古くて現実に合わない移動センサノードの位置を表示してしまうし、株価分析システムでは間違ったデータを元に株価を分析してしまう。それゆえ、これらのシステムにとって、データ鮮度を高めることは重要である。

データ鮮度を高めるには、新しいセンサデータがデータベースシステムに到着してからデータベースに挿入されるまでに要する時間を短くする必要がある。すなわちデータ挿入処理の高速化が必要とされる。その一方で、データ挿入処理結果は永続化されなければならない。

最も単純なデータ挿入処理手法は、挿入処理のたびに結果をディスクに書く方式である。しかしこの方式は遅いので、多くのデータベースシステムはバッファプールアクセス前にディスクにログを書くディスクロギング方式を採用している。

しかしディスクロギングはディスクアクセスを伴うために高速化には機械的な限界がある。そこで本研究ではネットワークを介して他ノードのメモリにログを書くメモリロギングに注目し、その速度と永続性を高める手法について提案をすると共に実験により評価をする。

本論文の構成は次のとおりである。2 節では挿入処理の高速化に関する関連研究について述べる。3 節ではセンサデータ挿入処理を高速化するシングルメモリロギングを評価する。4 節ではシングルメモリロギングの速度を向上させる方法を述べる。5 節ではメモリログの永続性を向上させる方法を述べる。6 節では 4 節と 5 節の結果をまとめ、メモリロギングの速度と永続性を向上させる、マルチメモリロギングを提案し、評価する。7 節では本研究の提案を改善する方法を述べる。8 節では本研究をまとめる。

2. 関連研究と本研究の寄与

2.1. 関連研究

2.1.1. Real-Time Data Server

Real-Time Data Server(RTDS) は、センサデータを獲得し、実時間で提供するデータサーバである [4][5][6][7]。

RTDS では、獲得されたデータは、時刻印を付与された後、獲得スレッドにより、主記憶上のリングバッファに獲得順に格納される。リングバッファが一巡した後は、新たなデータの獲得時には、最古のレコードが上書きされる。また、一定数の獲得が行われる毎に、退避スレッドによって、リングバッファ上の時系列データはディスクへ書き込まれ、永続化される。

文献 [4] において、RTDS ではリングバッファのサイズおよび退避のトリガとなる獲得回数を適切に設定することで、獲得スレッドと退避スレッドをそれぞれの時間制約を守るように動作させ、かつリングバッファ自身をロックすることなく、獲得されたデータを書き潰す前にディスクに書き込める、と述べられている。

しかし、RTDS のセンサデータ挿入処理方式には次の問題がある。

問題 1. リングバッファ上のデータが消失する可能性があること

RTDS では、新たに獲得されたデータをただちにディスクへ書き込まないために挿入処理が高速だと思われる。しかしディスクに書き込まれていないリングバッファ上のデータは、突発的電源切断時に消失してしまう。

それゆえ RTDS の挿入処理方式は、一部の消失が問題とならない種類のデータには適当だが、ひとつの消失も許されない種類のデータには不適当である。

問題 2. 非周期センサデータ挿入処理が難しいこと

RTDS では、ほとんどのセンサデータが周期的に発生することを前提としている。それゆえ金融データのように非周期的に発生するデータを扱う場合には、リングバッファ上のデータは、退避スレッドによりディスクに書き込まれる前に、獲得スレッドにより書き潰される可能性がある。

2.1.2. TimesTen

メインメモリデータベースシステム TimesTen[8] では、挿入処理を高速化するために、ログデータをディスクに書き込むタイミングを、ユーザが指定することができる。つまり、ユーザが指定したとき以外は、データベースシステムへの変更をディスクに書き込まないことで、データ挿入処理速度を向上させることが可能である。

この方式では、RTDS の問題 1 と同じく、突発的電源切断時に、主記憶上のデータが消失する可能性がある。ディスクに書き込まれる前のデータは永続化されてないので、電源が入ってもデータは復旧されない。

2.1.3. LazySync

Lazy synchronization(LazySync)[9] は、センサデータ挿入処理の高速化方式である。LazySync はセンサデータを主記憶にバッファリングし、クライアントが定める時間的一貫性制約を破らないタイミングで、バッファリングしていたセンサデータをディスクへ一括転送する手法である。LazySync は RTDS のセンサデータ挿入処理方式と似ている。しかし、時間的一貫性制約を破らないタイミングでバッファを一括転送する点が、RTDS のデータ挿入処理方式と異なる。

2.1.4. ClustRa

分散データベースシステム ClustRa[10] はロギング方式にメモリロギングを用いている。ClustRa のメモリロギングは、データ挿入時に、独立故障モードにある 2 サイトに対してレプリカ同期と合わせてログレコードを輸送する。しかし文献 [10] にはメモリロギングに用いるネットワークプロトコルも具体的な方式も記述されていない。

2.2. 関連研究のまとめと本研究の寄与

関連研究のデータ挿入処理の高速化手法まとめる。RTDS と TimesTen のデータ挿入処理方式および LazySync では、ディスクアクセスを高速化するために、ディスクへの書き込み操作を非同期的におこなう。しかし、そのアプローチでは、突発的電源切断時にメモリ上のデータを消失させるおそれがある。

それに対して ClustRa のアプローチは、メモリロギングにより、そのアプローチの問題であるデータの消失をなくし、かつデータ挿入処理を高速化する。データの消失を重視するならば、メモリロギングを用いるアプローチはディスクへの書き込み操作を非同期的におこなうアプローチよりも優れている。

しかし文献 [10] ではメモリロギングの速度と永続性について議論されていない。そこで本研究ではメモリロギングを評価し、その改善手法について述べる。

3. シングルメモリロギング

本節では、1 台のリモートノードのメモリにログを書くシングルメモリロギングを、実験によりディスクロギングと比較する。

図 1にシングルメモリロギングの概要を示す。

シングルメモリロギングの実行手順は次の通りである。

1. データベースシステムはセンサデータを受信

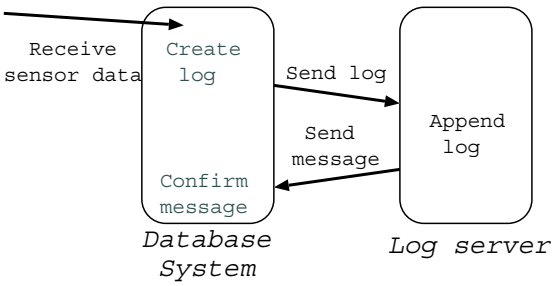

図 1: シングルメモリロギング概要

2. データベースシステムはログレコードを作成
3. データベースシステムはログレコードを送信
4. ログサーバは新しいログを追加
5. ログサーバは処理成功メッセージを送信
6. データベースシステムは処理成功メッセージを確認

ディスクロギングとシングルメモリロギングの速度を実験で比較した。実験環境、実験結果、そして考察を以下に述べる。

3.1. 実験環境

実験用データベースシステムを C 言語とアセンブラを用いて構築した。ディスクロギングシステム、シングルメモリロギングシステム、そしてデータベースシステム以外の環境について説明する。

3.1.1. ディスクロギングシステム

ディスクロギングシステムでは、バッファプールに新しいデータを挿入する前にローカルディスク上のログファイルにログを書き込む。この方式を図 2に示すシステムに組み込んだ。

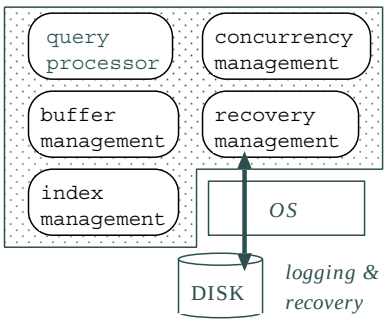

図 2: ディスクロギングシステム

クエリプロセッサで処理するオペレーションは、SELECT、INSERT、CREATE、DROP の 4 種類である。並行実行制御の方式は多版並行制御であり、データ挿入中もセンサデータ系列への読み出しアクセスが許可される。ま

た、1 クライアントに対して 1 プロセスを対応させる。各プロセスは共有メモリ上のバッファプールを排他制御しながらアクセスする。メモリ索引とディスク索引にはハッシュを用いた。センサデータ挿入用バッファプールの管理方式は、センサデータが追加されないことを考慮して、FIFO(First In First Out) を用いた。

3.1.2. シングルメモリロギングシステム

シングルメモリロギングシステムでは、バッファプールに新しいデータを挿入する前にリモートノードのログサーバにログレコードを渡し、ログサーバはメモリ上にログレコードを保管する。この方式を図 3に示すシステムに組み込んだ。

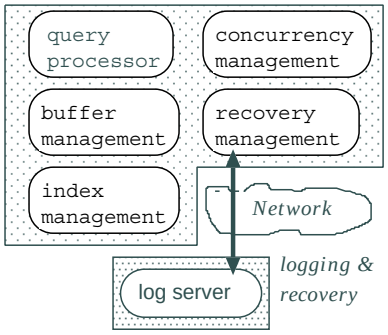

図 3: シングルメモリロギングシステム

このシステムの構成要素は、バッファマネージャとログサーバ以外はディスクロギングシステムと同じであるため、ここではバッファマネージャとログサーバについてのみ述べる。

- バッファマネージャ

  バッファプールが一杯になると、バッファマネージャはバッファプール内のデータをディスクに一括転送する。転送時にはバッファがロックされるが、ふたつのバッファを切り替えて使用することで、データ挿入処理がブロックされることを防ぐ。

  ディスクに転送されたデータは永続化されるから、それらのログレコードは不要になる。そこでデータベースシステムはログサーバにログレコード消去を依頼し、ログサーバは依頼されたログレコードを消去する。

- ログサーバ

  データベースシステムは新しいセンサデータ挿入以前にログレコードをログサーバに渡し、リカバリ時にはログサーバからログレコードを受け取って、故障時にメモリ上に存在していたデータを復旧する。

  ログサーバの動作は次のようになる。

  1. ログレコードをデータベースシステムから受信
  2. センサ名をキーとして挿入するログリストを探索
  3. そのログリストの末尾へログを追加
  4. 追加成功メッセージをデータベースシステムへ送信

データベースシステムからログサーバへ送られるログレコードの内容とサイズを図 4に示す。ログレコードが持つのは、コマンド、挿入するセンサ系列の名前、センサデータの値、センサデータがデータベースシステムへ到着した時刻である。コマンドには、ログレコード追加、ログレコード消去、そしてリカバリの 3 種類がある。

属性	説明
コマンド	4 バイト
センサ名	32 バイト
データ値	8 バイト (double 型)
到着時刻	8 バイト (timeval 型)
レコードサイズ	52 バイト

図 4: ログレコードのサイズ

3.1.3. 計算機環境

実験をおこなった計算機環境を表 1に示す。また、ネットワークプロトコルには、信頼性に問題のない TCP/IP を使用した。

表 1: 計算機環境

システム	詳細
OS	FreeBSD 4.4 RELEASE
ディスク	IBM-DTTA-350640
CPU	Pentium II 300MHz
メモリ	64MB
NIC	100-Mb Fast Ethernet
ケーブル	カテゴリ 5
ハブ	Fast ethernet switching hub

3.2. 評価

複数のセンサデータを同時にデータベースへ挿入するのに要する時間を、ディスクロギングとシングルメモリロギングについて測定した。ディスクロギングの方式には、ログファイルをセンサ数だけ用意する分散ログ方式と、ログファイルをひとつに集中する集中ログ方式を用いた。

センサデータを発生させるプロセスから、それぞれ 5000 件のセンサデータを同時に発生させ、それらのデータを

挿入するのに要した時間を測定した。5 回の測定結果から求めた 1 件あたりの挿入時間を表 2に示す。

表 2: ディスクロギングとシングルメモリロギングを用いた場合の一件あたりの挿入時間の比較

センサ数	ディスクロギング (ミリ秒)		シングルメモリロギング (ミリ秒)
	分散ログ	集中ログ	
1	1.400	1.414	0.374
2	2.298	2.404	0.536
3	3.480	3.658	0.834
4	18.544	4.806	0.968
5	38.616	5.970	1.148

表 2より、シングルメモリロギングは、分散ディスクログ方式に比べて最大で 33.6 倍速く、集中ディスクログ方式に比べて最大で 5.2 倍速いことがわかる。センサ数が 4 以上になると、シングルメモリロギングとディスクロギングの速度差は特に大きくなる。

この結果は、4 つ以上のプロセスが同時にディスクに書き込むことに、分散ログ方式が対応することが難しいことを表している。センサ数を 6 以上に増やして実験すれば、この差は一層広がるだろう。したがって、4 台以上のセンサから頻繁に新しいデータが到着する場合には、データベースシステムのロギング方式に、分散ディスクロギングを使うべきではない。

3.3. シングルメモリロギングの問題

シングルメモリロギングはディスクロギングより速いことがわかった。しかし上記のメモリロギング方式には 2 つの問題点がある。

ひとつ目は、1 つのノードにしかロギングしないために永続性が弱い点である。ログサーバが動作しているノードの電源が切れたら、メモリログレコードは失われてしまう。ふたつ目は、ネットワークプロトコルが重い点である。TCP は信頼性があるが遅い。データベースシステムとログサーバのネットワーク距離は高々1 ホップだから、TCP より処理が速いプロトコルが使用可能である。

これらの問題点を改善する手法を 4 節と 5 節で述べる。4 節では速度向上について、5 節では永続性向上について述べる。

4. シングルメモリロギングの速度改善

4.1. ネットワークプロトコルの処理速度

TCP は信頼性があるが遅い。そこでプロトコルを変えることにより速度改善を試みた。試した方法は (1)UDP の利用、(2)BPF(Berkeley Packet Filter) を経由するリンク層へのパケット投入である。プロトコル階層を図 5に示す。

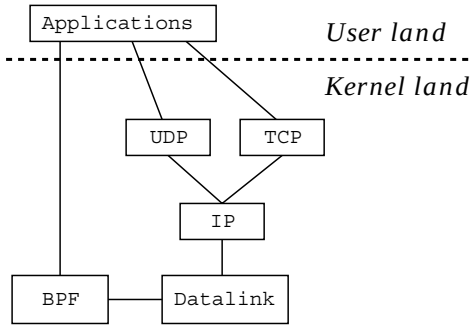

図 5: プロトコル階層

1. UDP

   TCP はコネクション確立、パケット到着保証、パケット到着順序確認、そしてフロー制御をおこなう。UDP はこれらを一切行わないために高速である。

2. BPF 経由通信

   TCP や UDP を用いると、トランスポート層とネットワーク層を通るあいだにチェックサム付加などの処理が行われる。これらをユーザプロセスであらかじめ処理しておき、データリンク層に渡せば高速化される可能性がある。

   そこで、BPF を経由してユーザプロセスからデータリンク層へ UDP パケットを投げる 機構を、libnet[11] ライブラリを利用して実装した。しかし受け取りは BPF 経由にせず、UDP を利用した。その理由は、libpcap[12] ライブラリを用いて BPF 経由でパケットを受け取る実験をしたところ、UDP を利用するよりも大幅に遅くなったからである。

以上の手法について、ラウンドトリップタイムを比較した。TCP、UDP、そして BPF 経由通信を用いた際の、50000 回のラウンドトリップタイムの平均を表 3に示す。ログレコードのサイズは図 4同様、52 バイトとした。

表 3: ラウンドトリップタイムの比較

プロトコル	ラウンドトリップタイム (ミリ秒)		
	最大	最小	平均
TCP	1.149	0.221	0.243
UDP	1.085	0.189	0.192
BPF 経由通信	1.153	0.195	0.193

表 3より、 UDP と BPF 経由通信は TCP よりもわずかに高速だといえる。しかし UDP と BPF 経由通信は TCP のような信頼性がなく、パケットが届かない事態に対処で

きないことが問題である。そこで次に UDP と BPF 経由通信に信頼性を与える手法を述べる。

4.2. ネットワークの信頼性

UDP と BPF 経由通信を用いた場合のパケット落ちに備えて、再送タイムアウト設定およびシーケンス番号の付加をおこなう。再送タイムアウトは 1.2 ミリ秒に設定した。その理由は、表 3で示されているように、UDP プロトコルを使ったときのラウンドトリップタイムが 1.2 ミリ秒以下だったからである。

文献 [13] では UDP に信頼性を付加する手法として、ラウンドトリップタイムに基づく動的な再送タイムアウト設定、シーケンス番号、そしてタイムスタンプの付加が述べられている。しかし、文献 [13] の対象はインターネットであるのに対し、本研究の対象はローカルエリアネットワークなので、パケットが故障する可能性は低い。さらに動的タイムアウト設定とタイムスタンプ付加は処理コストが大きい。それゆえ本研究では再送タイムアウトを固定し、シーケンス番号を付与するだけにした。

5. メモリログの永続性改善

5.1. ログサーバの増加

メモリログの永続性を改善するために、ログサーバを 1 台から 3 台に増やす。データベースシステムは 3 台のログサーバにログレコードを送り、3 台のログサーバからログ追加成功メッセージを受け取ることでロギングを終える。なお、ネットワークプロトコルには UDP マルチキャストを用いた。

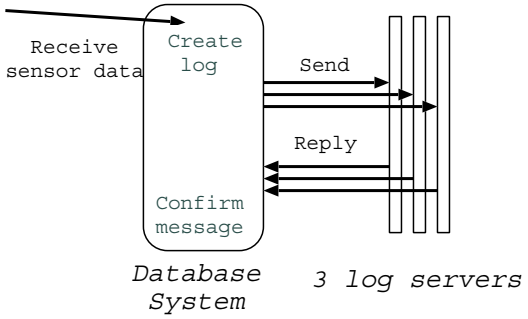

図 6: 3 台のログサーバへのロギング

5.2. ログサーバマネージャ

5.1 節で述べたロギング方式では、ログサーバが 1 台でも故障していればロギングを終了できない。そこでログサーバを管理するログサーバマネージャを実装した。

ログサーバの反応が無ければ、データベースシステムはログサーバマネージャに反応がないログサーバを知らせ、ログサーバマネージャから稼働中のログサーバを教わる。そして教わったログサーバにロギングを実行する。

5.2.1. 切り替え手順

データベースシステムがログサーバを切り替える手順を次に示す。

1. データベースシステムは同一ログサーバへログを 3 回送信
2. ログサーバから応答がなければ、データベースシステムはそのログサーバが故障していると判断し、ログサーバマネージャに故障ログサーバの IP アドレスとポート番号を通知し、新しいログサーバの情報を要求
3. ログサーバマネージャは、故障ログサーバを故障リストに追加し、新しいログサーバの IP アドレスとポート番号をデータベースに通知
4. 通知を受けると、データベースシステムは稼働中のログサーバのいずれかにログ同期依頼メッセージを送信
5. メッセージを受け取ったログサーバはログを新しいログサーバへ転送
6. 転送終了後、新しいログサーバは同期終了をデータベースシステムへ通知
7. データベースシステムはログ同期終了を受け取ると、新しいログサーバへロギングを実行

6. マルチメモリロギング

3 節で述べた速度改善案と 4 節で述べた永続性改善案より、ネットワークプロトコルには UDP を使用し、3 台のリモートノードにロギングを行う方式が、メモリロギングに望ましいといえる。以後、この方式を**マルチメモリロギング**と呼ぶ。

マルチメモリロギングを用いた場合に、複数のセンサデータを同時に挿入処理するのに要する時間を測定した。5000 件のセンサデータを挿入するのにかかる時間を 5 回測定し、平均値を計算した 1 件あたりの挿入時間を表 4に示す。

表 4: マルチメモリロギングによる挿入時間

センサ数	一件あたりの挿入時間 (ミリ秒)
1	1.230
2	1.590
3	2.274
4	2.884
5	3.616

そして分散ディスクロギング、集中ディスクロギング、シングルメモリロギング、そしてマルチメモリロギングの速度を比較するために、表2と表4をまとめ、図7に示す。

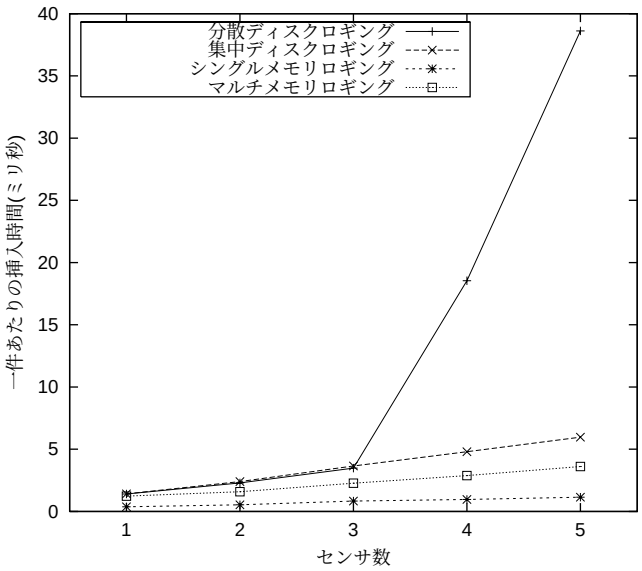

図 7: 挿入時間のまとめ

図7より、マルチメモリロギングはシングルメモリロギングより最大で約3.3倍遅いが、分散ディスクロギングより最大で約10.7倍、集中ディスクロギングより最大で約1.67倍速いことがわかる。

7. 改善案

本節では本研究で提案したマルチメモリロギングの速度向上とメモリログの永続性向上に関する改善案を述べる。

7.1. 速度向上について

メモリロギングの速度を向上させる案を以下に述べる。

1. マルチキャスト

   表2と表4より、マルチメモリロギングの速度は、シングルメモリロギングの約3倍になっていることがわかる。この理由は、ネットワークプロトコルにUDPユニキャストを用いたからだろう。UDPマルチキャストを用いることで速度向上が期待される。

2. デバイスドライバアクセス

   第4節においてメモリロギングの速度向上のためにBPF経由通信を試したが、UDPよりも遅くなった。しかしUDP/IPプロトコルスタック内で行われる処理をあらかじめユーザレベルで処理してEthernetフレームを作成し、それをデバイスドライバへ送れば、速度を向上できる可能性がある。

3. ログパック法

   マルチメモリロギングは、センサデータが到着する度に、52バイトのログ送信をおこなう。しかしEthernetのMaximum Transmission Unit(MTU)は通常1500バイトであり、Max Segment Size(MSS)は1460バイトなので、MSSに入りきる程度の複数ログを一括送信するログパック法を用いることで、ロギング速度を向上できる可能性がある。

   同様な手法をディスクに対して行う方式としては、前述のTimesTen[8]のログ同期方式と、LazySync[3]がある。

   ログパック法を用いると、これらの方法と同様に、突発的電源切断時にメモリ上のデータが消失する可能性があるので、クライアントの時間一貫性制約を破らないログパックサイズを選択する必要がある。

4. ロギング方式の変更

   本論文で示したマルチメモリロギングは、3台のログサーバへログを送信し、3台全てのログサーバからackを受信することで、ロギングを終了させた。しかし、ログサーバがn台あるときに、n台全てのログサーバからackを受信するには、時間を要する。

   そこでnよりも少ないm台のログサーバからackを受信した段階でロギングを終了させるプロトコルを導入すれば、マルチメモリロギングの速度を向上させられる可能性がある。

5. 高速スイッチの利用

   本研究で用いたEthernetのMACプロトコルは、CSMA/CDである。CSMA/CDは、ノードが増えるにつれて、データ衝突が発生する可能性も指数関数的に高くなると言われている。

   しかし、ハブにスイッチングハブを使えば、衝突はノードとスイッチの間では発生するものの、ノード間では発生しない[14]。それゆえスイッチングハブを使えば、Ethernetでも提案方式を拡張できる。

   しかし提案方式のボトルネックはスイッチ速度にあるので、RHiNET[15]のような高速スイッチを利用することが望ましい。

7.2. 永続性向上について

1. ログサーバマネージャの冗長化

   ログサーバマネージャが停止した場合、メモリロギングを実行するデータベースシステムはログサーバを切り替えることができず、処理がその時点で停止してしまう。したがってログサーバマネージャは常に多重化されることが望まれる。

2. 遠隔地へのロギング

3 台のログサーバにログレコードを渡しても永続性に不安が残る。停電が発生すれば、3 台のログサーバの電源が同時に切れるから、ログレコードは消失するからである。

この問題へのひとつのアプローチとして、遠隔地のログサーバへのロギングがある。この手法はラウンドトリップタイムがディスクロギングに比べて十分短ければ有効だと言える。そこで、研究室のホストから、研究室と同キャンパスの別棟にある Information Technology Center(ITC) のホストと、藤沢市遠藤にある湘南藤沢キャンパス (SFC) のホストへのラウンドトリップタイムを測定した。ネットワークプロトコルには TCP を用いた。この結果を表 5に示す。

ディスクロギングがもっとも速いのは、表2によると、センサ数がひとつの分散ロギングの場合である。このとき一件あたりのロギング時間は 1.4 ミリ秒である。ITC へのラウンドトリップタイムの平均値である 0.433 ミリ秒は、それに比べて約 3.23 倍速い。しかし SFC へのラウンドトリップタイムはディスクロギングよりも遅い。それゆえログサーバがクライアントにある程度近ければ、ロギング速度をディスクロギングよりも高めながら永続性を強化できる。

表 5: 遠隔地へのラウンドトリップタイム

サーバの場所	ラウンドトリップタイム (ミリ秒)		
	最大	最小	平均
ITC	10.417	0.415	0.433
SFC	343.516	1.792	1.868

7.3. 改善案のまとめ

7.1 節と 7.2 節で述べた改善案のうち、速度向上案 1、2、4 および永続性向上案 1 は必須である。いくつかのセンサデータが消えても許される場合には、それに速度向上案 3 を加えて良いだろう。

速度向上案 5 と永続性向上案 2 は同時には実行できない。速度向上が重視される場合には速度向上案 5 を採用し、永続性が重視される場合には永続性向上案 2 を採用すべきである。

8. まとめ

センサデータを扱うシステムにとって、データ鮮度を高めることは重要である。データ鮮度を高めるには、永続性を保ちながらデータ挿入処理速度を向上させる必要がある。

そこで本研究ではメモリロギングに注目した。まず、1 台のリモートノードのメモリにログを書くシングルメモリロギングを評価した。そしてシングルメモリロギングの速度と永続性を高める手法として、マルチメモリロギングを提案した。

マルチメモリロギングは、ログを 3 台のログサーバに対して UDP ユニキャストを用いて送信し、ack を受信する。実験の結果、マルチメモリロギングは集中ディスクロギングに比べて、最大で約 1.67 倍高速になることがわかった。そして最後にマルチメモリロギングの速度と永続性を改善する案を述べた。

高いデータ鮮度が求められる、株分析システムやヒューマンコンピュータインタラクションシステムの基盤となるデータベースシステムにはマルチメモリロギングが有効だろう。今後、マルチメモリロギングの速度と永続性を高める手法について研究をすすめる所存である。

謝辞 有益なコメントをくださった査読者の方に感謝します。

参考文献

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

[2] Michimune Kohno and Yuichiro Anzai. An Adaptive Sensor Network System for Complex Environments. In *Proceedings of 5th International Conference on Intelligent Autonomous Systems*, pp. 21–28, 1998.

[3] 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 Advanced Database Symposium*, pp. 105–112, December 2000.

[4] 井戸譲治, 島川博光, 竹垣盛一. 実時間データ獲得システム RTDS の評価. 情報処理学会研究報告データベースシステム, No. 109, pp. 159–164, July 1996.

[5] 高田秀志, 井戸譲治, 竹垣盛一. 有効期限に基づく時間的正当性を考慮した時間的オブジェクトモデルとその応用. 情報処理学会研究報告データベースシステム, No. 109, pp. 147–152, July 1996.

[6] 島川博光, 井戸譲治, 竹垣盛一. 時間制約と一貫性制約を満たす実時間データ獲得システム RTDS. 情報処理学会研究報告データベースシステム, No. 109, pp. 153–168, July 1996.

[7] 白川洋充, 竹垣盛一. リアルタイムシステムとその応用, 第 7 章. 朝倉書店, September 2001.

[8] http://www.timesten.com/.

[9] 川島英之, 遠山元道, 安西祐一郎. 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. 情報処理学会研究報告 2001-OS-86, 2001.

[10] Svein-Olaf Hvasshovd, Øystein Torbjørnsen, Svein Erik Bratsberg, and Per Holager. The ClustRa Telecom Database: High Availability, High Throughput, and Real-Time Response. In *Proceedings of 21th International Conference on Very Large Data Bases*, pp. 469–477, September 1995.

[11] http://www.packetfactory.net/projects/libnet/.

[12] http://www.tcpdump.org/.

[13] W. Richard Stevens. *UNIX NETWORK PROGRAMMING*, Vol. 1, chapter 20. Prentice Hall, second edition, 1998.

[14] RichSeifert 著, 間宮あきら訳. LAN スイッチング徹底解説, 第 4 章. 日経 BP 社, August 2001.

[15] 西宏章, 多昌廣治, 稲沢悟, 西村信治, 工藤知宏, 天野英晴. RHiNET スイッチ RHiNET-2,3/SW. 情報処理学会研究報告 2001-ARC-144-10, pp. 55–60, 2001.