センサデータベースシステムKRAFTの設計と実装

川島英之[†] 今井倫太[††,†††]
遠山元道[††] 安西祐一郎[††]

本論文ではセンサデータベースシステム KRAFT を提案する．KRAFT は様々なセンサデータを管理するとともに以下の 3 つの機能を提供する．(1) データの永続性と鮮度，(2) センサデータを管理する抽象型，(3) 連続的にセンサデータを監視する機能．(1) の機能を提供するために KRAFT はローカルディスクの代わりにリモートメモリを用いる．各ログレコードはリモートログサーバへ TCP を用いて並行的に送られる．リカバリ時にはリモートログサーバの全ログレコードが時間順に集められ，冪等にデータベースサーバへと送り戻される．(2) の機能を提供するために KRAFT は関係データモデルを拡張し，類似シーケンスを検索できるセンサ型を提供する．類似性の距離尺度としてはユークリッド距離とダイナミックタイムワーピング距離を提供する．(3) の機能を提供するために，KRAFT はデータベースサーバ内部で連続的にクエリを実行できるモニタを提供する．我々は KRAFT の有用性を，ヒューマノイドロボット Robovie を用いた実験を通して示す．

Design and Implementation of a Sensor Database System KRAFT

Hideyuki Kawashima,[†] Michita Imai,[††,†††] Motomichi Toyama[††]
and Yuichiro Anzai[††]

This paper proposes a sensor database system KRAFT. KRAFT is a database system that supports a variety of sensor data, and provides the following features: (1) freshness for sensor data without losing persistence, (2) abstract data type for sensor data, and (3) continual monitoring. To provide the feature (1), KRAFT uses remote memory as WAL files instead of local disk. Each log record is sent to a remote log server with TCP concurrently. At recovery, all of the log records at the remote log server are collected and sorted by ascending time order and sent back idempotently to the database server. To provide the feature (2), KRAFT extends relational data model, and permits users to make SENSOR TYPE to retrieve similar sequences with euclid or dynamic time warping metric and to retrieve the last sensor data. To provide the feature (3), KRAFT permits to make MONITOR that periodically executes query inside KRAFT. We have applied KRAFT for a humanoid robot called Robovie, and measured the staleness of sensor data. The results of experiments indicated that KRAFT provides good freshness of sensor data and good scalability.

1. はじめに

本論文では，インタラクション支援や実時間地震同定等のセンサデータ処理システムの運営や開発に必要となるセンサデータベースシステムについて取り上げる．

これらのシステムにおけるセンサデータの取扱いを考えると，オンラインで獲得される大量のセンサデータをいかに管理・処理するのかが問題となる．獲得されたセンサデータは，システム自体に用いられるだけでなく，システムの開発者によりシステムの調整のため，すぐに用いられる．さらに，システムの実行が終了した後に，システムの再設計のための情報としても用いられる．したがってセンサの利用形態として，オンラインとオフラインの双方が要求される．

センサデータを大量に用いたシステムは数多く研究されている．コンピュータやロボットによって人間の支援や自然環境の監視を行うためには大量のセンサデータを扱う必要がある．文献 1) では，看護師の負担を

† 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
School of Science for OPEN and Environmental Systems, Faculty of Science and Technology, Keio University
†† 慶應義塾大学理工学部情報工学科
Department of Information and Computer Science, Faculty of Science and Technology, Keio University
††† 科学技術振興機構さきがけ研究
Precursory Research for Embryonic Science and Technology, Japan Science and Technology Agency

減らすためにウェアラブルセンサを看護師に装着させ，看護記録を半自動化する試みが述べられている．文献 2)，3) では，人間と円滑なインタラクションが可能なコミュニケーションロボットを実現するために，大量の人間のモーションデータを解析する研究が述べられている．文献 4) では，人と人とのインタラクションにおける社会的プロトコルを解析するために，カメラ/マイク/生体センサを用いる研究が述べられている．

これらのシステムを支援するために，データベースシステムは次の 3 つの問題を解決する必要がある．問題 (1)：鮮度と永続性のあるセンサデータを与える機能を提供すること．この機能は，インタラクション支援システムの状態を監視するだけでなく，インタラクション終了後に様々な解析を行うオフラインの機能を実現する．問題 (2)：類似シーケンスを検索する機能を提供すること．なぜなら人間支援システムでは，連続するセンサデータをセグメント化し，類似したセンサデータのシーケンスを見つけることにより有益な情報が得られるからである．たとえば，類似した人間の動作を，異なる支援場面で検出されたとすると，それらの支援場面で人間に及ぼされた作用は類似していたと考えられる．問題 (3)：センサデータを周期的に監視する機能を提供すること．実験中にセンサデータを継続的に監視することは，インタラクション支援システムの開発者に実験状況を深く理解させるだけでなく，センサによる誤作動の発見を容易にする．

これらの問題に対して，すでに様々なデータベース技術が開発されてきた．問題 (1) を解くために，我々は以前リモートロギング法を提案した[5]．問題 (2) を解くために，様々な分野のアプリケーションが開発されている．一例をあげると，類似音楽シーケンスを検索するために，音楽検索システムである SoundCompass[6] が開発されている．問題 (3) を解くために，STREAM[7]，Telegraph[8]，Aurora[9] 等のストリームプロセッサが開発されている．

リモートロギング法はセンサデータストリームに有効であるが，これは問題 (2) も問題 (3) も解決しない．ストリームプロセッサは頻繁に到着するデータを監視し続ける機能を提供するが，データを永続化しないから問題 (1) を解決しないうえに，類似シーケンスを検索する処理を持たないため問題 (2) を解決する手段も提供しない．SoundCompass は音楽検索という点において問題 (2) を解決するが，特定用途にのみ有用であり一般的なデータを管理するシステムソフトウェアとして開発されていないうえに，問題 (1) と問題 (3) を解決する手段を提供しない．

表 1 Robovie のセンサ
Table 1 Sensors of Robovie.

センサの種類	センサの数
超音波距離センサ	24
全方位カメラ	1
温度センサ	1
赤外線センサ	2
触覚センサ	10
タッチセンサ	16

そこでデータベースシステムとして問題 (1)，(2)，(3) を同時に解決するために本論文ではセンサデータベースシステム KRAFT☆を提案する．問題 (1) を解決するために，文献 5) ではセンサデータにしか適用できなかったリモートロギング法を，関係データベースシステム内の拡張データ型であるセンサデータ型に適用できるように修正を施して KRAFT に組み込む．問題 (2) を解決するために関係データモデルにセンサ抽象データ型を導入したデータモデルを KRAFT に適用し，さらにそのセンサデータ型に対して類似シーケンス検索を実行できる問合せ手段を提供する．問題 (3) を解決するために周期的監視機構を KRAFT に導入する．

本論文の構成は以下のとおりである．2 章は KRAFT のターゲットアプリケーションの 1 つであり，ヒューマンロボットインタラクションの研究者に使われているコミュニケーションロボット Robovie を述べ，問題を定式化する．3 章は前述した 3 つの問題を解決する方法を述べる．4 章は 3 つの問題解決手法の同時実現方式として，なぜデータベースシステムに組み込む方式を選ぶのか，そしてその実現に際して何が問題かを述べる．5 章は KRAFT の設計と実装を述べる．6 章は評価実験と実行例を通して，KRAFT が問題を同時に解決することを示す．7 章は KRAFT の適用可能性と課題を述べる．最後に 8 章は結論を述べる．

2. Robovie と問題の定式化

2.1 Robovie

ロボットによる人間支援システムの研究をするために，ATR 知能ロボティクス研究所は Robovie-II[10] と呼ばれるコミュニケーションロボットを開発している．以下簡略のためにこれを Robovie と表記する．Robovie は**図 1** に示されるヒューマノイド型ロボットである．Robovie は 11 自由度を持ち，**表 1** に示す多数のセンサを持つ．

☆ **K**ernel of **R**eal-time **A**ctive and **F**resh **T**ime-series data manager

図 1 コミュニケーションロボット：Robovie
Fig. 1 Communication robot: Robovie.

ヒューマンロボットインタラクションの実験では，研究者はデータベースシステムに以下の 3 つの問題を解くことを求める．問題 (1)：センサデータの鮮度と永続性の実現．なぜなら彼らは実験中にロボットの状態を監視するだけでなく，実験後に様々な解析を行うからである．問題 (2)：類似シーケンス検索機能の提供．なぜなら彼らはセンサデータから一連のコンテキストを切り出すために，シーケンスの類似性を利用するからである．問題 (3)：センサデータの周期的監視機能の提供．なぜなら彼らは実験中にロボットの状態を継続的に監視するからである．

これら 3 つの問題はヒューマンロボットインタラクションにおいて求められるばかりでなく，1 章で述べた幅広い種類の人間支援システムでも求められる．

2.2 問題の定式化

上記 3 つの問題を本節で定式化する．

Problem1 （センサデータの鮮度と永続性） s をすでにコミットされたセンサデータとする．我々は s の古度と鮮度をそれぞれ次のように定義する．ただし，at(s) を s がデータベースシステムに到着した時刻とし，rt(s) を s がクエリ処理により読まれた時刻とする．

$$\mathsf{staleness(s)} = \mathsf{rt(s)} - \mathsf{at(s)} \tag{1}$$

$$\mathsf{freshness(s)} = \frac{1}{\mathsf{staleness(s)}} \tag{2}$$

この定義は staleness(s) が小さいほど freshness(s) は高まることを示す．そして，s が永続性を持つことを persistent(s) と記述することにする．ここで我々は第 1 の問題を次のように定式化する：「persistent(s) と高い freshness(s) を同時に提供すること」．

Problem2 （類似シーケンス検索） ユークリッド距離は正確な検索結果を求める際に要求される距離尺度である．長さ N である 2 つのシーケンスを S_A と S_B とすると，S_A と S_B のユークリッド距離は次のように定義できる．

$$\mathsf{Dist_{euclid}}(S_A, S_B) = \frac{\sqrt{\sum_{k=1}^{N}(S_{A_k} - S_{B_k})^2}}{N} \tag{3}$$

ユークリッド距離よりも曖昧な検索結果を求める際に使われる距離尺度としてダイナミックタイムワーピングがある．これを Keogh [11] に従ってダイナミックタイムワーピング距離を定義する．まず，要素が距離 $d(S_{A_i}, S_{B_j}) = (S_{A_i} - S_{B_j})^2$ である $N \times N$ のマトリックスを作成する．ただし $1 \leq i \leq N$，$1 \leq j \leq N$ である．ワーピングパス W はマトリックス $(1,1)$ から始まり，最もコストの低いルートを選びつつ，(N,N) で終わる．選ばれたマトリクス点の値を w_k として，ダイナミックタイムワーピング距離を次のように定義する．ただし w_k と w_{k+1} は縦横斜めのいずれかの方向について隣接しており，k が 1 つ増えたとき，i と j の少なくともいずれか片方は増加する．

$$\mathsf{Dist_{dtw}}(S_A, S_B) = \min\left\{\frac{\sqrt{\Sigma_{k=1}^{K} w_k}}{K}\right\} \tag{4}$$

正確な検索結果を求めるときには $\mathsf{Dist_{euclid}}$ を使えばよく，若干曖昧な検索結果を求めるときには $\mathsf{Dist_{dtw}}$ を使えばよい．文献 2) の著者らに聞いたところ，これらの 2 つの尺度があれば，彼らが開発するヒューマンロボットインタラクションシステムでの類似シーケンス検索には十分に有用性があるとのことである．これより，我々はこれらの尺度が他のシステムにもある程度有用だと考える．そこで我々は第 2 の問題を次のように定式化する：「S_A と類似する S_B を検索する手法を提供すること．ただし距離尺度は $\mathsf{Dist_{euclid}}$ と $\mathsf{Dist_{dtw}}$ を許す」．

Problem3 （周期的監視） 我々は周期 period で duration の期間にわたり実行するクエリ query を周期的監視 monitor とし，第 3 の問題を次のように定式化する：「monitor (period, query, duration) を提供すること」．

3. 問題解決方法

本章では，2 章で定式化した 3 つの問題を解くための方法を提案する．

3.1 問題 1：データ鮮度と永続性の両立

3.1.1 アプローチ

問題 1 を解くために，我々はリモートロギング法を KRAFT に適用する．一般的に，センサデータに優れた鮮度を与えることに対する最大の障壁は，遅いディスクアクセスである．この問題を解決するために，リ

モートロギング法ではリモートメモリにログレコードを置く．我々の以前の結果[5]では，リモートロギング法は 60 倍以上，ディスクへの Write-Ahead Logging（WAL）よりも平均鮮度が優れる結果が得られている．

リモートロギングに関して，我々の以前の手法[5]と本論文の手法には次の違いがある．(1) より複雑なデータ構造を扱っていること．(2) 確実なデータ転送のために TCP のみ用いること．(3) 以前の手法では実現できていなかった冪等なリカバリ方式を実現していること．次項でこれらのアルゴリズムを示す．

3.1.2 アルゴリズム

3.1.2.1 ロギングプロトコル

タプル追加処理（insert）とセンサデータオブジェクト追加処理（append）のためのロギングプロトコルを**図 2** に示す．他の操作コマンドはスペースのために省略する．2 行目において，ログサーバは新しいロギングスレッドを作成する．クライアント 1 つにつき 1 つのログスレッドが対応するので，クライアントとログスレッドの数は等しくなる．

6–11 行目では，タプル追加（insert）処理が記述されている．6 行目では，ログサーバはタプルフレームデータ構造（図 8 における “*TUPLE_FRAME*”）とタプルデータを受け取る．なぜならタプルデータ領域は動的に確保されるので，別々に送信される必要があるからである．

9–10 行目では，ログサーバはファイルの末尾オフセットを到着したタプルに与えたあと，そのファイル末尾オフセットを次のタプルに備えて増加させる．この処理は冪等なリカバリ処理を実現するために行われる．リカバリ時には各タプルに記録されているファイルオフセットを利用してリカバリ処理が行われる．

12–17 行目では，センサ追加（append）処理が記述されている．これは 6–11 行目と同じ処理である．

3.1.2.2 リカバリプロトコル

図 3 はリカバリプロトコルを示す．リカバリ処理は並行的には実行されない．したがってロギングよりも時間を要する．2 行目で，L はコミットされていないトランザクションのログレコードを破棄する．これらのレコードは各ロギングスレッドが保持するログレコードの末尾部分に存在する可能性がある．なぜならトランザクションは begin で始まり，commit で終了するからである．5 行目において，D はレコードからトランザクションを再構築する．

3.2 問題 2：類似検索

3.2.1 アプローチ

問題 2 を解決するために，KRAFT のセンサ型は類似シーケンス検索を許す．センサ型は $\mathsf{Dist}_{\mathsf{euclid}}(S_A, S_B)$ も $\mathsf{Dist}_{\mathsf{dtw}}(S_A, S_B)$ も許す．KRAFT は SQL のサブセットを拡張して，類似検索を行う．

たとえば，“(1,2,3)” に類似した，距離尺度が $\mathsf{Dist}_{\mathsf{euclid}}(S_A, S_B) < 10$ であり，表の名前が Robovie で，属性名が sonar1 のセンサデータシーケンスを検索するには，以下のように記述すればよい．

```
select * from Robovie where simseq external
Robovie.sonar1 [1,2,3] with euclid < 10.
```

このクエリでは，**simseq external** は “外部シーケンスと類似したシーケンスを検索せよ” を意味する．**with euclid** は “距離尺度を $\mathsf{Dist}_{\mathsf{euclid}}(S_A, S_B)$ にせよ” を意味する．もしも **euclid** が **dtw** と記述されて

```
1: C connects D, and D connects L.
2: L creates a logging thread for the connection.
3: C sends an operation to D.
4: D sends a log record to L.
5: switch (type of the log record) {
6: case insert:
7:    D sends tuple frame to L and L sends ack.
8:    D sends tuple content to L and L sends ack.
9:    L gives offset of file tail for the tuple
10:   L increments the offset for the tuple
11:   break;
12: case append:
13:   D sends sensor frame to L and L sends ack.
14:   D sends sensor content to L and L sends ack.
15:   L gives offset of tail for the sensor
16:   L increments the offset for the sensor
17:   break;
18: }
19: L sends ack to D.
20: D sends message to C.
```

C: client, D: database server, L: log server

図 2 ロギングプロトコル
Fig. 2 Logging protocol.

```
1: D starts and connects to L with recovery port.
2: L collects all of log records limiting in
   committed transactions from log threads.
3: L sorts the records by ascending time order.
4: L sends all of the sorted records to D.
5: D makes transactions from the records.
6: D executes redo of the transactions.
```

D: database server, L: log server

図 3 リカバリプロトコル
Fig. 3 Recovery protocol.

いれば，距離尺度を $\mathsf{Dist}_{\mathsf{dtw}}(S_A, S_B)$ にすることを意味する．

また，**external** は **internal** や **latest** に変えることもできる．**internal** は内部シーケンスとの比較を表し，**latest** は最新シーケンスとの比較を表す．

内部シーケンスとの比較は次のように行う．

```
select * from Robovie where simseq internal
Robovie.sonar1[1][0,0][1081238444,0] with euclid
= 0
```

Robovie.sonar1[1][0,0][1081238444,0] の意味は，“ID が 1 で，比較元シーケンスは UNIX 時間で 0 秒 0 μ 秒から，1081238444 秒 0 μ 秒” である．

最新シーケンスとの比較は次のように行う．

```
select * from Robovie where simseq latest 10
item of Robovie.sonar1[1] with euclid = 0
```

simseq latest 10 item of の意味は，“最新の 10 個のオブジェクトから構成されるシーケンスを比較元として類似検索を行う” である．これを **simseq latest 10 s of** と変更すれば，“最新 10 秒間のオブジェクトから構成されるシーケンスを比較元として類似検索を行う” になる．

3.2.2 アルゴリズム

類似シーケンス検索アルゴリズムを**図 4** に示す．このアルゴリズムは brute force 方式である．すなわちスライディングウィンドウを先頭から終端まで移動させていく間に，ユーザが指定した条件を満たす窓があれば，それを答えリストに追加していく．1 行目でウインドウがセットされる．2 行目から 7 行目において検索処理が実行される．空間索引等はまだ準備されていないので，検索コストは高い．

3.3 問題 3：周期的監視

3.3.1 アプローチ

問題 3 を解くために，KRAFT は周期的監視を作成することをユーザに許可する．以下にその例を示す．

```
create monitor sonar1_monitor period 1 s as se-
lect ∗ from Robovie where latest Robovie.sonar1
```

“**create monitor**” は新たな周期的監視を作成することを意味する．**period 1 s** は周期的監視のクエリ周期が 1 秒であることを意味する．**latest** は Robovie.sonar1 の最後のデータのみ得ることを意味する．周期的監視 sonar1_monitor を実行するために，

```
1: Set W(Q);
2: for (i := 0; i < M; i++) {
3:   Calculate d using W(Q) and W(S_i);
4:   If (d meets condition(e.g, =0, <10, etc.)) {
5:     Creates a new SENSOR_SEGMENT;
6:     Copies W(S_i) to it;
7:   }
8: }
```

Q: Query sequence of length n
S: Sequence in database constituted of M elements
S_i: Subsequence of S starting i-th element of length n
d: Distance between Q and S_i
W: Window for Q or S_i

図 4 サブシーケンス検索アルゴリズム
Fig. 4 Algorithm for subsequence retrieval.

ユーザは次のようなコマンドを発行する．

```
run sonar1_monitor during 300 s
```

“**run**” は周期的監視を呼び出すことを意味する．**during 300 s** は周期的監視の実行期間を 300 秒に設定することを意味する．期間が過ぎると周期的監視は停止する．

3.3.2 アルゴリズム

周期的監視を実現するには 2 種類の最も単純なアルゴリズムがある．1 つのアルゴリズムは，周期の期間中に一度だけ実行を行い，残りはずっと眠り続ける，怠惰なアルゴリズムである．このアルゴリズムの長所は負荷が軽いことであり，短所は鮮度の高いデータを読めないことおよび周期を超えてしまう可能性があることである．もう 1 つのアルゴリズムは，周期の期間中ずっとエグゼキュータを呼び出し続ける，貪欲なアルゴリズムである．このアルゴリズムの長所は鮮度の高いデータを読めて周期を超える可能性も低いことであり，短所は負荷が重いことである．

そこで我々はこれら 2 つのアルゴリズムの組合せを用いる．これを**図 5** に示す．1–2 行目において，WT と TDL はそれぞれ wait time と tentative deadline を表す．WT はスリープする時間である．TDL は周期の終わりに対する仮想的なデッドラインであり，デッドラインミスを防ぐために必要である．WT と TDL は次のように計算される．

$$\mathsf{WT} = period \times \alpha \quad (0 < \alpha < 1) \tag{5}$$

$$\mathsf{TDL} = period - \beta \quad (0 < \beta < period) \tag{6}$$

パラメータ α と β により，freshness(s)，デッドラインミス率，システムへの負荷は変動する．freshness(s)

```
1: result := invoke executor and measure time;
2: calculate WT and TDL;
3: sleep(WT);
4: while (now is before TDL) {
5:   result := invoke executor;
6:   sleep(γ);
7: }
8: send result to client;
```

図 5 周期的監視のアルゴリズム
Fig. 5 Monitor algorithm.

が変動する理由は，周期が終わるまでに，複数のセンサデータ追加処理が発生する可能性があるからである．

3 行目において，周期的監視は WT だけスリープする．そして 4–7 行目において，TDL が過ぎるまで何度もエグゼキュータを呼び出す．このループの中で，γ だけのスリープを行う．最後に 8 行目において，クエリ実行結果がクライアントに返却される．

パラメータ α，β，γ はシステム管理者により調整される必要があるが，我々はこれらを自動調整する機構を導入する予定である．

4. 問題解決方法の同時実現

センサデータ処理システムの運営者や開発者は，基本的な関係データベースシステムの機能をデータ管理ソフトウェアに求めるため，彼らの要求を満足するためには，2 章で述べた 3 つの問題を解決しながら，関係データベースシステムの機能を同時に提供する必要がある．

この技術的問題点を解決するにあたり，3 機能をデータベースシステムに組み込む方式（組み込み方式）と，単純に 3 機能とデータベースシステムを組み合わせる方式（ラッパ方式）が考えられるが，ラッパ方式の性能は組み込み方式よりも劣る．その理由を以下に述べる．

4.1 ラッパ方式の性能が組み込み方式より劣る理由

4.1.1 類似シーケンス検索についての比較

ラッパ方式で類似シーケンス検索を実現すると次のような処理が行われる．(1) ユーザからのクエリをラッパが受け取り，(2) それをパーザが解析して内部表現および SQL クエリに変換し，(3) データベースシステムに SQL クエリを発行してその結果を取得し，(4) ラッパ内で類似検索を実行し，(5) そして最後に処理結果をクライアントに返却する．ここで負荷が大きいのは (3) である．なぜなら (3) ではデータベースからラッパへクエリ結果を転送するからである．このとき結果を転送するには時間がかかるうえ，転送中には転送データを送信側と受信側で保持するためメモリも必要になる．

組み込み方式では (3) は不要なのでラッパ方式は組み込み方式より多くのメモリと長い処理時間を必要とする．それゆえ類似シーケンス検索の性能は，ラッパ方式よりも組み込み方式が優れる．

4.1.2 リモートロギング法についての比較

ラッパ方式でリモートロギング法を実現するには，センサデータ用のログをリモートロギングシステムで管理し，関係データ用のログをデータベースシステムで管理し，それらを束ねる統合ログマネージャを開発する方法が考えられる．この方法をとるならば，リカバリとチェックポインティング時に統合ログマネージャが両ログを合わせて統合ログを作成し，それを用いてリカバリもしくはチェックポインティングを実行する．

組み込み方式では統合ログを作成する必要はないから，統合コストは存在しない．それゆえリモートロギング法の性能は，ラッパ方式よりも組み込み方式の方が優れる．

4.1.3 周期的監視機能についての比較

ラッパ方式で周期的監視機能を実現するには，監視処理の呼び出しごとにデータベースシステムにクエリを発行し，プロセス間通信により結果を取得する必要がある．

組み込み方式ではこの処理は不要なため，周期的監視機能の性能はラッパ方式よりも組み込み方式が優れる．

4.2 組み込み方式を実現するうえの課題

以上の議論より，組み込み方式はラッパ方式よりも優れた性能を提供することが分かった．組み込み方式の実現はラッパ方式とは異なり，各機能の単純な組合せでは実現できない．その実現にはデータベースシステム内の様々なモジュールに修正を施す必要がある．組み込み方式を実現するために必要な課題を以下で述べるとともに，各課題の解決への参照を示す．

4.2.1 類似シーケンス検索についての課題

組み込み方式で類似シーケンス検索を実現するには 2 つの問題がある．

第 1 の問題はシーケンスデータを保持するセンサ型を関係データベースシステムに導入する必要があることである．この解決のためには，新たなデータモデルを考案し，関係データベースシステムのメモリ管理とストレージ管理を修正する必要がある．センサ型を導入するデータモデルは 5.1.1 項で述べる．メモリ管理は 5.3.1 項で述べる．ストレージ管理は 5.3.2 項で述

べる．第 2 の問題は，類似シーケンス検索要求を解釈するようパーザを修正し，検索処理をエグゼキュータに含めることである．パーザは 5.2.1.2 項で，エグゼキュータは 5.2.1.3 項で述べる．

4.2.2 リモートロギング法についての課題

組み込み方式でリモートロギング法を実現するには，関係データベースシステムのための WAL 処理を，文献 5) に示されているリモートロギング法に修正する必要がある．そのため，ロギング実行処理・リカバリ処理・チェックポインティング処理を修正する必要がある．

これらの解決のうち，ログセンダについては 5.2.1.4 項で，トランザクションマネージャについては 5.2.1.6 項で，そしてログサーバについては 5.2.2 項で述べる．リカバリの解決は 5.2.1.5 項で述べる．チェックポインティングの解決は 5.2.2.3 項で述べる．

4.2.3 周期的監視機能についての課題

組み込み方式で周期的監視機能を実現するには，クエリの答えをクライアントに転送する時間によって監視処理の周期をずらされない手法が導入されなければならない．たとえば周期が 1 秒の監視処理において，クエリ結果のクライアントへの転送に 2 秒かかると，監視処理が周期的に動作できない．それゆえ周期的監視を行うスレッドと結果転送を行うスレッドを作成し，それらをつなぐ機構が必要である．そしてパーザにも修正が必要となる．

この解決は，周期的監視の実現については 5.2.1.1 項で述べ，パーザの修正は 5.2.1.2 項で述べる．

5. 設計と実装

本章では KRAFT のデータモデルおよびアーキテクチャに関する設計と，実装を述べる．

5.1 データモデル

ここでは KRAFT のデータモデルの設計およびその設計の必然性を述べる．

5.1.1 データモデル設計

KRAFT ではすべてのデータは論理的にテーブルとして見える．KRAFT が提供するデータ型は，整数，実数，日付，固定長テキスト，センサ整数型，そしてセンサ実数型である．この中で，センサ整数型とセンサ実数型が KRAFT 独自の新しいデータ型である，センサ型である．これらセンサ型の要素であるセンサデータオブジェクトは，センサデータがデータベースサーバに到着した時刻と，センサデータ値の組からなる．センサ型は複数個のセンサデータオブジェクトを保持する．KRAFT は複数個のセンサデータオブジェクトをシーケンスと見なし，そこから類似するサブシーケンスを切り出す手段を SQL の拡張として与える．

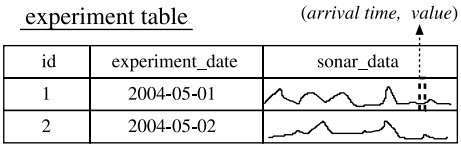


図 6 KRAFT のデータモデル
Fig. 6 Data model of KRAFT.

表 2 KRAFT が提供する操作
Table 2 Operations supported by KRAFT.

操作コマンド	内容詳細
(create|drop) table	Create or drop a table
(create|drop) monitor	Create or drop a monitor
select	Retrieve data from database
insert	Insert a new tuple into table
append	Append a new sensor data into sensor attribute
delete	Delete tuples
update	Update tuples

図 6 はロボット実験に用いるテーブル例を示している．属性は整数型の id，日付型の experiment_date，そしてセンサ整数型の sonar_data からなる．メモリマップの実装については 5.3.1 項で述べる．

5.1.2 操作インタフェース

表 2 は KRAFT が提供する操作を示す．このうち，**append** 操作は KRAFT 特有で，新しいセンサデータオブジェクトを追加するために使われる．たとえば，experiment というテーブルの中の，id が 1 であるタプルに値が v のセンサデータを追加するには，“append into experiment.sensor_data values (v) where id = 1” と記述する．

select 操作では，選択・射影そして結合をサポートするが，入れ子になった select はサポートしない．結合演算のアルゴリズムは入れ子ループ結合である．

5.1.3 データモデルの必然性

データベースシステムのデータモデルは，ユーザにとって使いやすく，さらに優れた性能を提供する必要がある．たとえば，KRAFT のユーザである Robovie 実験者は，センサデータを実験日や実験名と関連付けて管理することをデータベースシステムに求める．これを満足するモデルとして，センサデータを表現する属性をテーブル内に入れるデザインは直感的で分かりやすい．これを KRAFT データモデルと呼ぶ．

他方，関係データモデルでデータを表現しようとすると，テーブル ID・時間そしてデータを属性として持つテーブルと，テーブル ID・実験日そして実験名

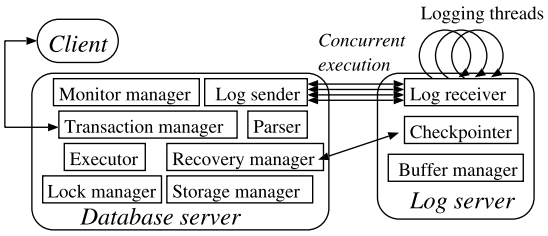


図 7 KRAFT のアーキテクチャ
Fig. 7 Architecture of KRAFT.

を持つテーブルを用意しておき，検索時にそれらを結合演算でまとめることで関連付けを行うことになる．センサごとにテーブルを作ると多くの結合演算を実行する必要があるため処理コストがかかるうえ，直感的にデータ間の関連を理解しにくい．さらに類似演算の実行はデータベースシステムからセンサデータを受け取った後になるため，KRAFT データモデルでは不要な通信コストが必要になる．

それゆえユーザにとって使いやすい操作インタフェースと優れた性能を提供するためには，関係データモデルでなくて KRAFT データモデルを用いることは必然であった．

5.2 アーキテクチャ設計

図 7 は KRAFT のアーキテクチャ設計を示す．KRAFT は 2 つのサーバから構成される．それらはデータベースサーバとログサーバである．データベースサーバはクエリをクライアントから受け取り，それを処理して，結果をクライアントに返す．ログサーバは通常のローカル WAL ファイルの代わりにログレコードを管理し，データベースサーバとインタラクションする．この実現にはリモートロギング法[5] を適用しているが，データベースシステムに導入するにあたり，複雑なデータ構造の適用と冪等なリカバリ処理を新たに実現した．このアルゴリズムは 3.1.2.2 項で述べた．

5.2.1 データベースサーバのモジュール構成

ここではデータベースサーバが持つモジュールを説明する．

5.2.1.1 モニタマネージャ

組み込み方式で周期的監視機能を実現するにはモニタマネージャが必要である．しかし既存のデータベースシステムにこのモジュールは存在しないため，我々はこれを新たに開発した．

このモジュールは各モニタの周期とクエリ内容を保持するとともにその実行も管理する．実行はパイプライン方式で行われる．すなわち，1 つのモニタについて 2 つのスレッドが起動される．一方のスレッドは監視結果を作成し，それをスレッド内部のキューに挿入する．もう片方のスレッドはキューから結果を取り出してクライアントに送る．この実行機構は，処理を分割することで監視処理の周期がずれることを防ぐように設計された．

5.2.1.2 パ ー ザ

組み込み方式で周期的監視機構と類似シーケンス検索を実現するには，それらの要求を解釈できるようにパーザが修正される必要がある．

このモジュールは SQL ライクなクエリで発行される類似シーケンス検索・周期的監視の作成と実行そしてセンサデータ追加コマンドである **append** を解釈して内部表現に変換できるように開発された．

5.2.1.3 エグゼキュータ

組み込み方式で類似シーケンス検索を実現するには，エグゼキュータを修正する必要がある．類似シーケンス検索として $\mathsf{Dist}_{\mathsf{euclid}}$ と $\mathsf{Dist}_{\mathsf{dtw}}$ を実現し，さらに不等号および等号を処理する機能を追加した．

5.2.1.4 ログセンダ

組み込み方式でリモートロギング法を実現するには，データベースサーバからリモートログサーバへログレコードを転送するモジュールであるログセンダが必要である．既存のデータベースシステムにこのモジュールは存在しないため，我々はこれを新たに開発した．

このモジュールは作成したログレコードを TCP プロトコルに則りログサーバへ転送し ack を受け取る．

5.2.1.5 リカバリマネージャ

組み込み方式でリモートロギング法を実現するには，リモートログサーバと通信してログレコードを受け取りリカバリ処理を実行するモジュールがデータベースサーバ内に必要である．既存のデータベースシステムにこのモジュールは存在しないため，我々はこれを新たに開発した．

このモジュールはリカバリ処理を実行する．データベースサーバが起動して初期化を済ませた後，最初に呼び出すのはこのモジュールである．このモジュールはリモートログサーバからログレコードを受け取り，redo 処理を冪等に実行する．冪等な redo 処理は 3.1.2.2 項で述べた．

5.2.1.6 トランザクションマネージャ

組み込み方式でリモートロギング法を実現するには，トランザクションを管理するモジュールが，クライアントとリモートメモリマネージャの両方に対するコネクションを管理する必要がある．なぜならクライアントがデータベースサーバに接続したときに，それに対応するスレッドがデータベースサーバとリモートメモリサーバの両方で 1 つずつ起動するからである．

このモジュールはスレッドの情報に加えてクライアントとのコネクションを管理する．クライアントがデータベースサーバに接続すると，このモジュールはその情報を内部リストに登録し，それからリモートログサーバに接続して，リモートログサーバ内で新たなロギングスレッドを作成するように依頼する．クライアントがデータベースアクセスを終了するときには，このモジュールはログサーバに関連するロギングスレッドを消去するように依頼し，そして内部リストから当該クライアントの情報を消去する．

5.2.1.7 ロックマネージャ，ストレージマネージャ

組み込み方式を用いる場合，これらのモジュールを既存のデータベースシステムから修正する必要はない．ロックマネージャは並行実行制御を管理する．並行実行制御プロトコルは 2 相ロックであり，シリアライゼーションレベルは read committed である．ストレージマネージャはデータをローカルディスクに書き込む．このモジュールが呼び出されるのはリカバリおよびチェックポインティングである．

5.2.2 リモートログサーバのモジュール構成

リモートログサーバは 3 つのモジュールから構成される．それらはバッファマネージャ・ログレシーバ・そしてチェックポインタである．これらのモジュールは既存のデータベースシステムに存在しないうえ，KRAFT では文献 5) に書かれている手法と異なりセンサ型以外の関係データも扱うため，新たに設計・開発した．

5.2.2.1 バッファマネージャ

バッファマネージャは，ファーストフィットに従ってメモリ割当てを行う．バッファが足りない場合にはチェックポインタが呼び出される．このモジュールは必要不可欠である．なぜならメモリが足りなくなればロギングができなくなるため，メモリ不足時にはそれを解決する手段を自前で持たなければならないからである．

5.2.2.2 ログレシーバ

ログレシーバは，ログクライアントのためにロギングスレッドを作成する．したがってロギングは並行的に実行される．ロギングスレッドはログクライアントからログレコードを受け取って管理する．

5.2.2.3 チェックポインタ

チェックポインタはすべてのログレコードを回収し，リカバリマネージャにそれらを渡して，データの永続化を行う．この処理は冪等に実行される．このモジュールは必要不可欠である．なぜならメモリ不足を解決する手段であるからである．

5.2.3 モジュール構成の必然性

5.2.3.1 類似シーケンス検索に関するモジュール

類似シーケンス検索機能をデータベースシステムに組み込むために，我々がパーザとエグゼキュータを修正した理由を述べる．パーザはユーザから受け取ったクエリを内部表現に変換するモジュールだから，従来のテーブル操作クエリだけでなく，3.2.1 項で述べた類似シーケンス検索クエリも解釈する必要がある．その要求を実行する類似検索実行部をエグゼキュータ外部のモジュールとして開発することも不可能ではないが，エグゼキュータに組み込む方が設計が単純で美しくなり，エグゼキュータと外部モジュール間のインタラクション消去による性能改善も見込めるため，我々は類似検索実行部をエグゼキュータに組み込んだ．

それゆえ我々が類似シーケンス検索機能をデータベースシステムに組み込むためにパーザとエグゼキュータを修正したことは必然的だった．

5.2.3.2 リモートロギング法に関するモジュール

リモートロギング法をデータベースシステムに組み込むために，我々がログセンダをデータベースサーバに組み込む一方で，ログレシーバ・チェックポインタ・そしてバッファマネージャをログサーバに組み込んだ理由を述べる．

リモートロギング法では新しいログをローカルディスクには書かずにリモートホストのメモリに書くため，ログをデータベースサーバからログサーバに転送する必要がある．この転送を実現するためにログセンダはデータベースサーバに組み込まれ，ログレシーバはログサーバに組み込まれる．

チェックポインティングはログデータ領域へのアクセスを禁じつつ，ログレコードから復元したデータをデータベースサーバ上のディスクに書き込む処理であるから，チェックポインタはログデータ領域を持つログサーバに組み込まれることが性能面から見て好ましい．

ログサーバはメモリが不足するとログを置けなくなるため，メモリ管理を OS に委ねるわけにはいかない．すなわちログサーバが使うメモリはログサーバが管理し，不足時にはチェックポインタを呼び出す必要があるため，ログサーバにバッファマネージャが組み込まれる．

それゆえ，我々がログセンダをデータベースサーバに組み込み，ログレシーバ・チェックポインタ・そしてバッファマネージャをログサーバに組み込んだことは，リモートロギング法を組み込み方式で実現するために必然的だった．

5.2.3.3 周期的監視機能に関するモジュール

周期的監視機能をデータベースシステムに組み込むために，我々がモニタマネージャをデータベースシステムに組み込んだ理由を述べる．

モニタマネージャをデータベースシステム内に配置しなくても周期的監視は実現できる．しかし配置がデータベースシステムから遠ざかるにつれて監視処理の性能は劣化し，監視周期の間隔は不正確になる可能性が高まる．そこでエグゼキュータを呼び出せる程度の深度にモニタマネージャを配置することで，一般的なクエリに対応すると同時に性能向上を図った．

それゆえ KRAFT がモニタマネージャをデータベースシステム内部に配置したことは，性能とクエリの多様さを両立する周期的監視を実現するために必然的だった．

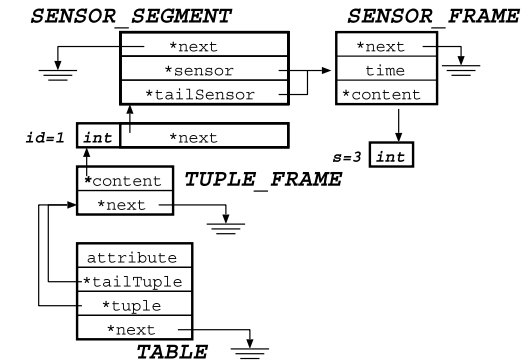
5.3 実　　装

KRAFT は第 1 著者によりスクラッチから開発された．KRAFT はすべて C 言語により記述されており，行数は 18,000 程度である．KRAFT は pthread ライブラリを利用しており，Linux と FreeBSD 上で動作する．

5.3.1 データベースサーバのメモリマップ

データベースサーバのメモリマップは，通常のデータベースシステムとは若干異なる．なぜなら KRAFT はセンサ型を持つからである．タプルはリストで実現され，データエリアはタプルのサイズに応じて動的に確保される．センサ型がタプルに存在する場合には，センサ型のポインタがタプルエリアに割り当てられる．そして各センサデータオブジェクトはリストにより実現される．

メモリマップの例とそれを作成するコマンドを**図 8** に示す．この中で，*TABLE* は *TUPLE_FRAME* を指している．*TUPLE_FRAME* 中の ∗content が指すのは，タプルのデータを保持する領域である，タプルデータオブジェクトである．図 8 の場合，タプルデータオブジェクトは INT 型データオブジェクトと，*SENSOR_SEGMENT* 型ポインタオブジェクトから構成される．この INT 型データオブジェクトの値は 1 であり，*SENSOR_SEGMENT* 型ポインタオブジェクトの ∗next は，動的に確保される *SENSOR_SEGMENT* 型データオブジェクトを指す．*SENSOR_FRAME* 型ポインタオブジェクトの ∗sensor は，動的に確保される *SENSOR_FRAME* 型データオブジェクトを指す．*SENSOR_FRAME* 型データオブジェクト内の ∗content はセンサデータオブジェクトを指す．この場合，その型は INT で値は 3 である．




```
% create table experiment (id int, sonar_data sensor_int)
% insert into experiment values (1, null)
% append into experiment.sonar_data values (3)
where id = 1
```

図 8 KRAFT のメモリマップ
Fig. 8 Memory map of KRAFT.

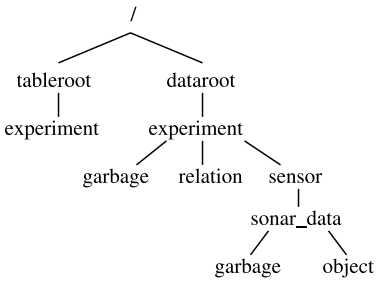


図 9 KRAFT のストレージ構成
Fig. 9 Storage organization of KRAFT.

5.3.2 データベースサーバのストレージ構成

図 9 は図 8 中に記述されたコマンドを用いたときに作成されるテーブルのストレージ構成を表す．

ディレクトリ “/dataroot/experiment/” は 2 つのファイル，“garbage” と “relation”，およびディレクトリ “sensor/” を持つ．“relation” はセンサデータ以外のタプルデータを持ち，“garbage” はすでに消去されたタプルのファイルオフセットを持つ．データベースサーバがローカルディスクからデータを読み込むとき，“garbage” に記述されたファイルオフセットは読み込まれずパスされる．

ディレクトリ “/dataroot/experiment/sensor/” は “sonar_data/” というディレクトリを持つ．そしてこのディレクトリには “garbage” と “object” というファイルがある．“object” ファイルはセンサデータオブジェクトを保持し，“garbage” ファイルはすでに消されたオフセットを持つ．

表 3 staleness(s) 測定実験の条件
Table 3 Conditions for staleness measurement experiment.

Number of appender	1
Period of appender	10 ms or 50 ms
Period of monitor	1 s
Duration of monitor	300 s

ファイル “/tableroot/experiment” は “experiment” テーブル内の属性情報を含む．

6. 評価実験と実行例

本章では KRAFT がいかにして 3 つの問題を解決するかを，評価実験と実行例を通して述べる．

6.1 KRAFT が問題 1 を解決すること

実験環境として，スイッチングハブで結合された 3 台のホストを用いた．ネットワークの帯域は 100 MB イーサネットである．ホストはそれぞれデータベースサーバ，ログサーバ，そしてクライアントに用いた．データベースサーバマシンの仕様は，CPU がクロック周波数 3 GHz の Pentium 4，メモリ 4 GB，そして OS が Linux Kernel-2.4.21-4 である．ログサーバマシンの仕様は，CPU がクロック周波数 2.4 GHz の Pentium 4，メモリ 1 GB，そして OS が Linux Kernel-2.4.21-4 である．クライアントマシンの仕様は，CPU がクロック周波数 2.0 GHz の Pentium 4，メモリ 1 GB，そして OS が Linux Kernel-2.4.21-4 である．各ホストにおいて PTHREAD_MAX_THREADS は 256 に設定されていたため，それ以上のスレッドは同時に作成できなかった．

6.1.1 staleness(s) の測定

我々は Robovie データの staleness(s) を，**表 3** に示した条件で測定した．まず我々は 11 のモータと 24 のソナーを持つ，Robovie という名前の表を KRAFT 内に作成した．それから周期的監視を作成した．それから，Robovie.motor1 にセンサデータを追加するアペンダスレッドを 1 つだけ作成して，周期的監視が得た staleness(s) を測定した．周期的監視の作成に用いたコマンドを**図 10** に，アペンダの実行に用いたコマンドを**図 11** に示す．

図 12 は staleness(s) を測定した結果を示す．この図の横軸は周期的監視処理の試行回数を表し，縦軸は staleness(s) を示す．アペンダの周期は 10 ms であるから，10 ms 以下の staleness(s) は，センサデータの永続化に遅延をまったく及ぼさないことになる．α，β，γ はそれぞれ 0.8，20 ms，1 ms と設定した．

図 12 において，試行を重ねるにつれて staleness(s) は徐々に上がって行き，10 ms を超えるとほとんど 0 ms まで低下する．その後はまた 10 ms へ近付いていくと考えられる．我々がこのように考えた理由は，アペンダ周期を 50 ms にした場合に**図 13** のような結果が得られたからである．図 13 からは staleness(s) が 50 ms へ近付くと 0 ms 近辺に低下する繰返しが観察される．

staleness(s) が徐々に変動する理由はデータ挿入処理と周期的監視の動作機構の違いにあると我々は考えている．データ挿入処理過程は次のようになる．(1) クライアントは sleep から目覚める．(2) クライアントはセンサ値を読み取りセンサデータを作成する．(3) そのデータが KRAFT データベースサーバに到着す

```
% create table Robovie (id int, motor1 sensor_double, ..., motor11 sensor_double, sonar1 sensor_double, ..., sonar24 sensor_double)
% insert into Robovie values (1, null, ..., null)
% create monitor monitor_Robovie period 1 s as select * from Robovie where latest Robovie.motor1
% run monitor_Robovie during 300 s
```

図 10 周期的監視の実行に用いたコマンド
Fig. 10 Commands for executing continual monitor.

```
while (1) {
  append into Robovie.motor1 values (1) where id = 1;
  sleep(period of appender);
}
```

図 11 アペンダの実行に用いたコマンド
Fig. 11 Commands for executing appender.

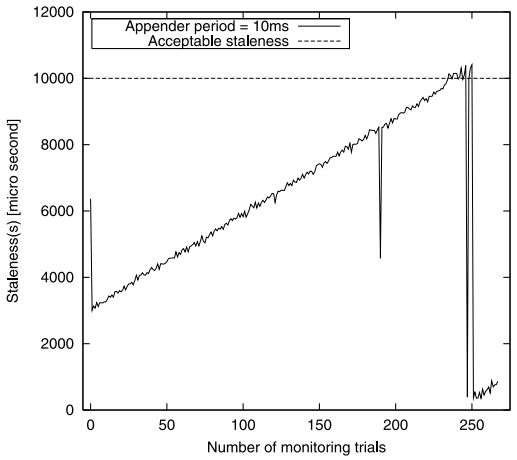


図 12 staleness(s) の遷移過程（アペンダ周期=10 ms）
Fig. 12 Transition process of staleness(s) (Appender period=10 ms).

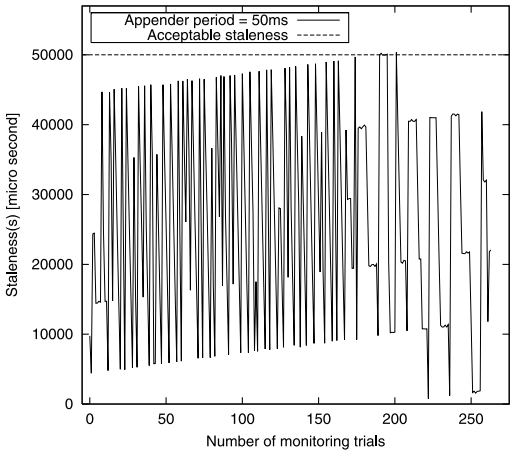


図 13 staleness(s) の遷移過程（アペンダ周期=50 ms）
Fig. 13 Transition process of staleness(s) (Appender period=50 ms).

る．(4) リモートログが実行され，データはバッファ領域におかれる．(5) 最後に ack がデータベースサーバからクライアントへ返される．(6) クライアントはそこで周期だけ sleep する．つまり (2)〜(5) に要する時間が変動のずれになりうる．このおよその時間を計算する．追加時間を表す図 15 において，32 並行度での 100 回の挿入時間は約 330 ms だった．したがって 1 並行度での 1 回の挿入時間は 300 μs 程度になり，これが (2)〜(5) に要する時間となる．以上よりセンサのスリープ周期が 10 ms の場合，センサデータが KRAFT に到着する周期はおよそ 10.3 ms になっていると考えられる．一方，周期的監視は 5.2.1.1 項で述べたパイプライン機構により，データ読み取り処理自体は変動なく周期的に実行される．それゆえ両者の動きの違いが変動ずれを生んだと我々は考察する．

図 12 において，周期である 10 ms を超えた staleness(s) は全体の 4.1%存在した．定義より staleness(s) が小さいほど freshness(s) は優れるため，この実験結果は KRAFT が優れた freshness(s) を提供することを示している．より詳細な実験結果を**表 4** に示す．表 4 から，staleness(s) の平均値は，ほぼ周期の半分となっており，さらに周期超え率は最悪でも 4.10%であることが分かる．

staleness(s) の平均値がセンサ周期よりも低ければ，最新のセンサデータを読めていたことになるため，平均値がセンサ周期の半分となることは，persistent(s) を得るためのコストをほとんどかけずに，最新センサデータを提供できたことを意味する．平均値が周期の半分になった理由は，図 12 より，staleness(s) が徐々に変動するからだと考えられる．

表 4 staleness(s) に関する統計的結果
Table 4 Statistical results with staleness(s).

センサ周期 (ms)	周期越え率 (%)	最悪値 (ms)	平均値 (ms)
10	4.10	10.4	6.23
20	2.25	23.9	11.2
30	1.13	30.4	16.1
40	0.38	94.1	21.6
50	1.13	50.4	26.3
60	0.00	58.2	29.7
70	1.50	76.4	35.7
80	1.12	80.4	43.0
90	0.37	90.6	44.7
100	0.00	98.6	50.3

```
1: アペンド数を X とする.
2: スレッドの ID を myid とする
3: for (i = 0; i < X; i ++) {
4:   append into X.myid values (i) where id = 1;
5: }
```

図 14 センサデータ追加プログラム
Fig. 14 Program to append sensor data.

以上より我々は，KRAFT が persistent(s) と優れた freshness(s) を同時に提供し，それゆえ問題 1 を解決すると主張する．

6.1.2 センサデータ追加時間の測定

staleness(s) に関する実験で KRAFT が優れたパフォーマンスを示した理由を理解するために，我々はセンサデータ追加処理時間を測定した．この実験では，我々は多数のアペンダスレッドを作成して並行的に実行させた．各スレッドには一定数のセンサデータ追加処理を発行させ，その合計実行時間を測定した．これをプログラムで示すと**図 14** のようになる．実験では，X の値を 100 から 500 までに設定した．最大並行度を表す myid の最大値は 32 から 224 まで測定した．

図 15 は実験結果を示す．図 15 の中で，X 軸はスレッド 1 つあたりの総データ追加処理数を表し，Y 軸は全スレッドが終了するまでにかかった時間を表す．KRAFT が優れた並行性を示すことを理解しやすくするために，並行度が 32 の場合に対して，何倍の時間を要したかを示す結果を**図 16** に示す．図 16 は，並行度が N 倍になっても実行時間が N 倍より低くなっていることを明らかに示している．それゆえ KRAFT は優れた並行性を示しているといえる．

図 16 には，2 つの興味深い点が存在する．それらは，追加処理数が 100 で並行度が 128 の場合と，追加処理数が 100 で並行度が 224 の場合である．ここ

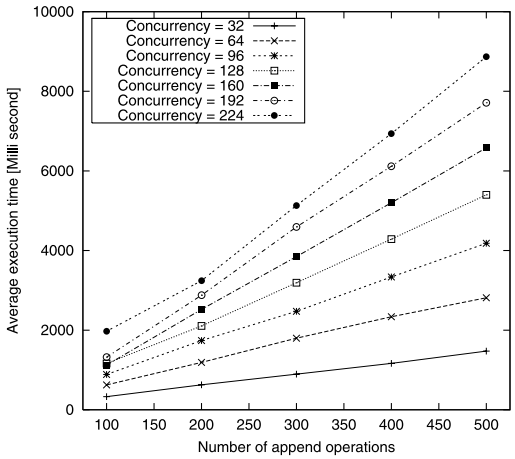


図 15 平均実行時間
Fig. 15 Average execution time.

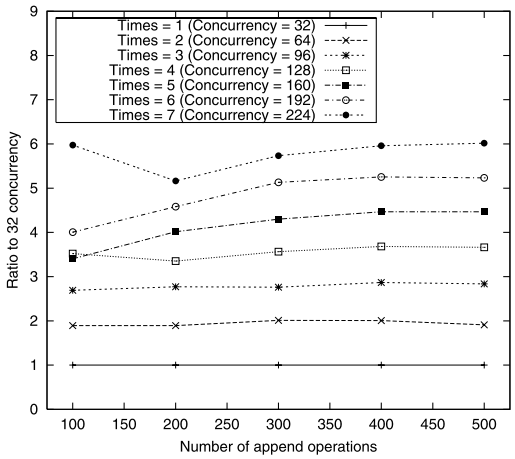


図 16 32 並行度との比較
Fig. 16 Comparing speed with case of 32.

で性能が悪化している理由は，*pthread_create* もしくはデーモンプロセスの影響を受けたことではないかと我々は推測している．

6.2 KRAFT が問題 2 を解決すること

我々は，KRAFT への類似シーケンス検索例を 3 つ示す．

- select sonar1 from robovie
  → $\{100, 110, 120, 130, 100, 170\}$
- select sonar1 from robovie where simseq external [100,100,100] with euclid $<$ 10
  → $\{100, 110, 120\}$
- select sonar1 from robovie where simseq external [100,100,100] with dtw $<$ 10
  → $\{100, 110, 120\}\{110, 120, 130\}\{120, 130, 100\}$

これらのサンプルクエリに関する $\mathsf{Dist}_{\mathsf{euclid}}$ と $\mathsf{Dist}_{\mathsf{dtw}}$ を**表 5** に示す．この結果より，$\mathsf{Dist}_{\mathsf{euclid}}$ は $\mathsf{Dist}_{\mathsf{dtw}}$ よりも堅い尺度であることが分かる．

表 5 サンプルクエリへの $\mathsf{Dist}_{\mathsf{euclid}}$ と $\mathsf{Dist}_{\mathsf{dtw}}$
Table 5 $\mathsf{Dist}_{\mathsf{euclid}}$ and $\mathsf{Dist}_{\mathsf{dtw}}$ for samples queries.

Subsequences in database	$\mathsf{Dist}_{\mathsf{euclid}}$	$\mathsf{Dist}_{\mathsf{dtw}}$
{100,110,120}	7.45	4.47
{110,120,130}	12.5	8.00
{120,130,100}	12.0	9.17
{130,100,170}	25.4	17.4

これらの実行結果より，KRAFT が $\mathsf{Dist}_{\mathsf{euclid}}$ と $\mathsf{Dist}_{\mathsf{dtw}}$ のいずれの距離尺度も許す類似検索を実行できると我々は主張する．それゆえ KRAFT は問題 2 を解決する．

なお実行時間は，アルゴリズムが brute force であるから長い．それゆえ索引や並列化による高速化が今後の課題である．

6.3 KRAFT が問題 3 を解決すること

6.1.1 項の実験より，KRAFT が monitor (period, operation, duration) を提供することは明らかである．それゆえ KRAFT は問題 3 を解決する．

7. KRAFT の適用可能性と有用性

本論文で我々は KRAFT がコミュニケーションロボット Robovie に対して適用可能であり有用であることを示したが，KRAFT は 2 章で述べた 3 つの機能をデータベースシステムに求めるアプリケーション全般に適用できる．たとえば，ロボット–センサネットワークおよび実時間地震同定システムはデータベースシステムに対して 3 機能を要求するため，KRAFT は適用可能である．ただし有用性は完全でない．これを以下で述べる．

7.1 適 用 例

7.1.1 ロボット–センサネットワークへの適用

我々は現在 MOTE[12] と複数台の Robovie を用いてロボット–センサネットワークを構築している．この目的は人間がインタラクションしやすい環境の開発である．

システム開発者は時時刻刻と変化するロボット–センサネットワークの状況をすぐに見たいため，データ鮮度と周期的監視機構が要求される．そして，オンラインまたはオフラインにおいてセンサデータの類似性から類似したインタラクションを同定し，それをもとに一連のインタラクションのメカニズムを解析するため，データの永続性と類似シーケンス検索機能が要求される．

ただし，現在の KRAFT は画像センサ型をサポー

トしないため，カメラから得られる画像センサデータを扱うことができない．

7.1.2 実時間地震同定システムへの適用

実時間地震同定システムでは，1 日に数 GB 程度得られる地震データを実時間に解析して，地震伝播パターンを予測する．このシステムに KRAFT を適用するために我々は地震研究機関と共同研究を始めたところである．

得られたばかりのセンサデータはオンライン解析に使われるため鮮度が求められる．さらにそのデータはデータベースに蓄えられて将来の解析時に検索される可能性があるため，永続化される必要がある．また監視は周期的に実行されることが求められるため周期的監視機能が要求される．予測アルゴリズムの 1 つとして過去の類似地震波の検索という手法があるため類似検索機能が要求される．

ただし，予測アルゴリズムは類似地震波検索以外にも自己回帰モデルへのあてはめ等の他の手法もあるため，実時間地震同定システムが要求するすべての予測手法を KRAFT が提供するわけではない．

7.1.3 適用可能性と有用性

上記 2 つのシステムには，データ鮮度，データ永続性，周期的監視機能，類似シーケンス検索が要求される．それゆえ，機能 (1)：データの永続性と優れたデータ鮮度，機能 (2)：類似シーケンス検索機能，機能 (3)：周期的監視機能が求められるため，ロボット–センサネットワークと実時間地震同定システムに KRAFT は適用可能であり，ある程度有用だともいえる．

しかし有用性は完全ではない．そこで有用性を増すために，KRAFT はロボット–センサネットワークにおいてカメラを支援するための画像型を提供し，実時間地震同定システムが求める類似シーケンス演算以外の解析手段を提供する必要がある．

7.2 今後の課題

上で述べたように，KRAFT はセンサデータ型とセンサデータシーケンス解析機能の豊富さにおいて有用性が完璧だとはいえない．さらに S-Plus や Mathematica をはじめとする時系列データ解析専用のソフトウェアは類似演算以外の様々な解析手法を提供するが，KRAFT はそれらを持たない．

そこで KRAFT の応用範囲を広げるべく，我々は今後の研究において，センサデータシーケンスに対する解析手段を強化するとともに，画像センサ型を導入する．求められる時系列データ解析手段はアプリケーションに応じて異なるため，我々は様々な場所でアプリケーションの要求をヒアリングしている．その過程で要求された機能を速やかに導入していく．また性能面においては，索引・並列処理・そしてデバイスコンシャスなアルゴリズムを用いた検索処理の高速化と，スレッドスケジューリングによる優れた実時間性の実現に向けて開発を進めている．

8. 結　　論

本論文ではセンサデータ処理システムを支援するために，センサデータベースシステム KRAFT を提案した．そして KRAFT は次の 3 つの問題を解決するデータベースシステムであることを示した．(1) データの永続性と優れたデータ鮮度を同時に提供すること．(2) 類似シーケンス検索を実行する手法を提供すること．(3) 周期的監視機能を提供すること．今後の課題はデータ解析手段の豊富化と性能向上である．

謝辞　本研究の一部には情報処理振興協会（IPA）による，平成 14 年度未踏ソフトウェア創造事業未踏ユース，プロジェクト名「センサデータベース管理システムの開発」のご支援をいただいた．IPA および PM の竹内郁雄電気通信大学教授に感謝する．慶應義塾大学の向井淳氏・佐竹聡氏・鳴海真里子氏・大澤幸子氏・根本潤氏・広瀬健志郎氏そして嶋村勇志氏には校正に関するコメントをいただいた．査読者の方々には貴重なご意見とコメントをいただいた．ここに記して謝意を表す．

参 考 文 献

1) 桑原教彰，野間春生，鉄谷信二，萩田紀博，小暮　潔，伊関　洋：ウェアラブルセンサによる看護業務の自動行動計測手法，情報処理学会論文誌，Vol.44, No.11, pp.2638–2648 (2003).
2) Imai, M. and Narumi, M.: Generating common quality of sense by directed interaction, *Proc. 12th IEEE International Workshop on Robot and Human Intaractive Communication* (*RO-MAN 2003*), pp.199–204 (2003).
3) 神田崇行，今井倫太，小野哲雄，石黒　浩：人–ロボット相互作用における身体動作の数値解析，情報処理学会論文誌，Vol.44, No.11, pp.2699–2709 (2003).
4) 角　康之，伊藤禎宣，松口哲也，シドニーフェルス，間瀬健二：協調的なインタラクションの記録と解釈，情報処理学会論文誌，Vol.44, No.11, pp.2628–2637 (2003).
5) 川島英之，遠山元道，今井倫太，安西祐一郎：リモートメモリを用いたセンサデータストリームの永続化，情報処理学会論文誌：データベース，Vol.44, No.SIG12 (TOD19), pp.98–109 (2003).

6) 小杉尚子，櫻井保志，森本正志：ハミング検索のための音楽データ自動時間正規化手法，情報処理学会論文誌：データベース，Vol.55, No.SIG7 (TOD22), pp.163–177 (2004).
7) Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J.: Models and Issues in Data Stream Systems, *ACM Symposium on Principles of Database Systems* (2002).
8) Madden, S.R. and Franklin, M.J.: Fjording the Stream: An Architecture for Queries over Streaming Sensor Data, *18th International Conference on Data Engineering* (2002).
9) Carney, D., Cetinternel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N. and Zdonik, S.: Monitoring Streams — A New Class of Data Management Applications, *Proc. 28th International Conference on Very Large Data Bases* (2002).
10) Kanda, T., Ishiguro, H., Ono, T., Imai, M. and Nakatsu, R.: Development and Evaluation of an Interactive Humanoid Robot "Robovie", *Proc. IEEE International Conference On Robotics and Automation*, pp.1848–1855 (2002).
11) Keogh, E.J. and Pazzani, M.J.: Scaling up Dynamic Time Warping for Datamining Applications, *Proc. 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.285–289 (1999).
12) XBOW. http://www.xbow.com

(平成 16 年 6 月 20 日受付)
(平成 16 年 10 月 4 日採録)

(担当編集委員 佐藤 哲司)

川島 英之（学生会員）

平成 11 年慶應義塾大学理工学部電気工学科卒業．平成 13 年同大学大学院計算機科学専攻前期博士課程修了．現在同大学院開放環境科学専攻博士後期課程在学中．データベースシステムの研究に従事．ACM 学生会員．

今井 倫太（正会員）

平成 4 年慶應義塾大学理工学部電気工学科卒業．平成 6 年同大学大学院計算機科学専攻修士課程修了．同年 NTT ヒューマンインタフェース研究所入社．平成 9 年 ATR 知能映像通信研究所へ出向．平成 14 年慶應義塾大学大学院理工学研究科開放環境科学専攻後期博士課程修了．現在，慶應義塾大学理工学部情報工学科専任講師および ATR 知能ロボティクス研究所客員研究員，科学技術振興機構さきがけ研究員．博士（工学）．VR 上のエージェントや自律ロボットとのインタラクションの研究に従事．ロボットとの対話，センサを用いた状況知覚に興味を持つ．電子情報通信学会，人工知能学会，ヒューマンインタフェース学会，日本認知科学会，ACM，IEEE 各会員．

遠山 元道（正会員）

昭和 54 年慶應義塾大学工学部管理工学科卒業．同大学大学院修士課程を経て昭和 59 年博士課程修了．同年同大学助手．現在情報工学科専任講師．博士（工学）．平成 8 年〜9 年オレゴン大学院大学（OGI）客員研究員．平成 10 年〜13 年 JST さきがけ研究員．主にデータベースの研究に従事．電子情報通信学会，IEEE Computer Society，ACM 各会員．

安西祐一郎（正会員）

昭和 49 年慶應義塾大学大学院博士課程修了．昭和 63 年より慶應義塾大学理工学部教授，平成 5 年より理工学部長．平成 13 年より慶應義塾長．この間昭和 56 年〜57 年カーネギーメロン大学客員助教授．計算機科学，認知情報処理過程の研究に従事．工学博士．電子情報通信学会，日本認知科学会，ACM，IEEE 等会員．