複雑時系列データのためのOLAPシステムの並列化手法

Parallelization of OLAP System to Analyze Complex Time Series Data

生田 泰章[*1] 猪口 明博[*1*2] 鷲尾 隆[*1]
Hiroaki Ikuta Akihiro Inokuchi Takashi Washio

[*1]大阪大学 産業科学研究所
The Institute of Scientific and Industrial Research , Osaka University

[*2]科学技術振興機構さきがけ
PRESTO, Japan Science and Technology Agency

In recent years, The storage and management of large amount of data are easily achieved with development of information technology. Much of the stored data is event time series having complex hierarchy, such as patients' medical histories. Under this situation, HealthCube have been developed to analyze very complex data by using interactive queries. However, HealthCube requires much computation time to respond to complex queries, because it is designed by using ROLAP. This paper proposes a method to reduce the computation time of HealthCube by parallelizing its processes.

1. はじめに

今日，計算機環境やネットワーク技術，ミドルウェア等の基盤技術の発展によって，大規模なデータの蓄積・管理ができるようになった．その中の多くの蓄積データは，複雑な構造を持った時系列データである．複雑な時系列データの例の 1 つとして，医療データが挙げられる．医療データは患者の診察や検査データ，投薬，手術などの項目からなる時系列データであり，それぞれの項目が複雑な階層構造を有している．

このような複雑な構造を持つ時系列データを分析する方法の一つに，多次元データベースがある．多次元データベースは分析の視点である次元を用いて，時系列データを対話的に分析することが可能である．しかし，従来の多次元データベースでは時間的順序関係を考慮した分析が困難であることや，多対多のリレーションシップを持つデータの保持が困難なことなど，課題が残されている [Pedersen 1999].

多対多のリレーションシップを多次元データベースに保存可能なスキーマとして BioStar スキーマ [Wang 2005] があるが，データの格納方法を中心に述べられており，時間的順序関係を考慮した分析などの処理方法について述べられていない．多次元データベースを拡張した OLAP システムである HealthCube[高林 2007] は，これら 2 つの課題を克服した手法を採用している．HealthCube は複雑時系列データをより詳細に分析し得る有用なシステムであるが，システム設計に ROLAP(Relational On-line Analytical Processing) の形態を採用しているために次のような課題が残されている．ROLAP は分析問い合わせを受けてから集計を行うので，大規模なデータに対してより対話的に分析するためには，応答時間の短縮が必要となる．そこで，本研究では蓄えられたデータを複数の計算機に分散させ，HealthCube の処理を並列化することで応答時間の短縮を図る．

2. 関連研究

本研究に関連する研究として，文献 [Chen 2004] がある．文献 [Chen 2004] では OLAP システムの並列化について，work partitioning と data partitioning の 2 種類に大別した．work partitioning は生データ 1 つに対する処理タスクを分割する．そして，複数の計算機がタスクを並列化し，結果を導く．一方，data partitioning はデータを各計算機に分散させ，検索に必要なデータだけを計算機から選択し，統合することで結果を導く．また，文献 [Chen 2004] は ROLAP の data partitioning の手法を提案している．具体的には，各計算機に分散されたデータに対して，各計算機が集計を行い，最終の結果の統合時にのみ計算機間の通信を行う．この手法は汎用性が高く，様々なデータについて有効な手法であるが，HealthCube のような時間的順序を考慮したデータモデルに対応していない．よって，本稿では，data partitioning 法を用いて HealthCube のデータを複数の計算機に分散させ，HealthCube のデータスキーマと処理手順に特化した並列化を提案する．

3. HealthCube

HealthCube は電子カルテ解析のために開発された OLAP システム [高林 2007] であり，以下の特徴を持つ．本節では，区間データとデータ操作演算を用いたクエリの統一的な処理手順について説明する．

- データを事象の開始時間と終了時間の情報を持つ区間情報として取り扱うことで，時間的順序の扱いを容易にするデータモデルとテーブルスキーマ．
- 分析者の様々な問い合わせを統一的に扱うことのできるデータ操作演算．
- 分析者の分析意図を直感的に表現でき，分析を容易に実行することを可能にするユーザインターフェース．

HealthCube はデータを区間情報としてテーブル I に蓄積する．ある患者の診療履歴を例に取ると，図 1 に示すように 2008 年 11 月 1 日 15 時から 2008 年 11 月 1 日 16 時までの間，胸部の CT を撮影するという検査をしていたとする．HealthCube のファクト表であるテーブル I に検査に関する区間情報が保存され，検査の開始時刻である 2008 年 11 月 1 日 15 時，検査の終了時刻である 2008 年 11 月 1 日 16 時が保存される．さらに，値には胸部 CT が保存され，その階層構造をもつカテゴリとして，ルートノードから CT に至るパスである検査/放射線/CT が付与される．なお，階層構造をもつカテゴリの情報はテーブル T に保存されている．

連絡先: 生田 泰章，大阪大学 産業科学研究所，大阪府茨木市美穂ケ丘 8-1，ikuta@ar.sanken.osaka-u.ac.jp

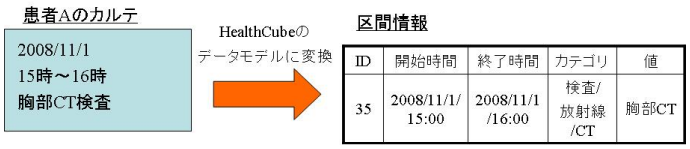

図 1: 区間情報の例

また，HealthCube は分析者の分析問い合わせをデータ操作演算を用いて以下に示す統一的な手順で処理し，結果を返す．以下に HealthCube で用いるデータ操作演算の概要を示す．

集計演算 α**:** テーブル $A = \{v_1, v_2, \cdots, v_n, v_{n+1}\}$ に対して，$\alpha(A, \sigma) = {}_{v_1,v_2,\cdots,v_n}\chi_{v_1,v_2,\cdots,v_n,\sigma(v_n+1)}$ を返す集計演算を $\alpha(A, \sigma)$ と定義する．ここで ${}_{a}\chi_{a,sum(b)}$ は SQL “select a, sum(b) from ... group by a” に相当する演算とする．また，$\sigma \in \{count, count\ distinct, sum, avg, max, min\}$ である．演算 α はテーブル A から，n 次元の多次元キューブを生成する関数である．

結合演算 β**:** 結合演算 β をテーブルの集合 $\{I'_1, I'_2, \cdots, I'_n\}$ に対して，$\beta(\{I'_1, I'_2, \cdots, I'_n\}, O, W) = \pi_O(I'_1 \bowtie I'_2 \bowtie \cdots \bowtie I'_n)$ を実行する結合演算と定義する．ここで，各テーブル I'_i は区間を表すタプルからなり $I' = \{id, start, end, value, interval\_id\}$ とする．また W は結合の条件式の集合であり，$I'_i \bowtie I'_j$ は，W の条件式，及び $I'_i.id = I'_j.id$ に従って結合される．O は出力される属性の集合であるとする．

区間選択演算 g**:** 区間選択演算 $g(T, I, c)$ を区間を表すテーブル I' を返す演算と定義する．ここで T，I，c はそれぞれ，カテゴリの情報を保存しているテーブル，区間データを保存しているテーブル，区間を選択する際に使用するカテゴリである．関数 g は分析の意図に応じて定義されるユーザ定義関数 [Inokuchi 2006] であり，テーブル I から分析者の望むカテゴリに属するデータを選択する．

HealthCube では保存されたテーブル I からデータ操作演算を用いて，以下に示す統一的な処理手順で分析者のクエリを処理する．

1. 区間選択演算 g を行い，集計に必要な区間を選択する．
2. 結合演算 β を行い，g により選択されたテーブルを ID の値と結合条件 W に従って結合する．
3. β により結合した結果に対し，集計演算 α を行う．

このような統一的な処理手順は式 (1) のように表現できる．

$$\boldsymbol{\alpha(\beta(\{g(T, I, c_1), g(T, I, c_2), \cdots\}, O, W), \sigma)} \qquad (1)$$

簡単な例を使って述べると，式 (2) のように分析者が HealthCube に，術式と CT 検査を受けた患者の術式ごと，検査部位ごとの患者数を得るクエリを発行したとする．

$$\boldsymbol{\alpha(\beta(\{g(T, I, \text{“術式”}), g(T, I, \text{“}CT\text{”})\}, O_1, W_1),\ \text{‘}count(distinct\ id)\text{’})} \qquad (2)$$

このとき，図 2 のように，まず区間選択演算 g により，術式の情報と CT 検査の情報の 2 つの区間を選択する．ここで，区間選択演算 g により選択された術式情報からなるテーブルを I'_1，CT 検査情報からなるテーブルを I'_2 とする．その後，結合演算 β により，結合条件 W と $I'_1.id = I'_2.id$ に従って，I'_1 と I'_2 を結合する．ここまでの処理で，どの患者がどの術式とどの部位の CT 検査をしたかを得ることが出来る．最後に，集計演算 α により患者数の集計を行い術式と CT 検査のクロス表を返す．本研究では，HealthCube の演算の処理効率が良く

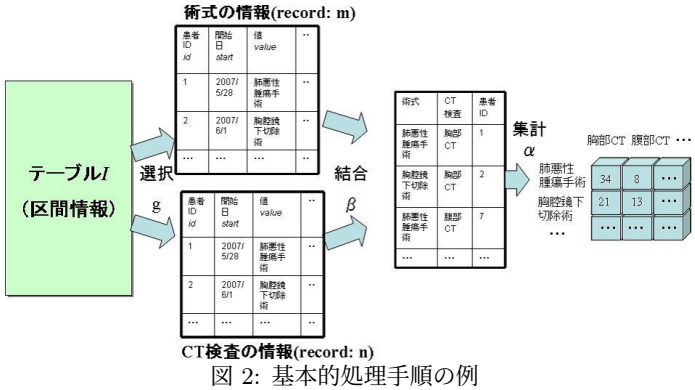

図 2: 基本的処理手順の例

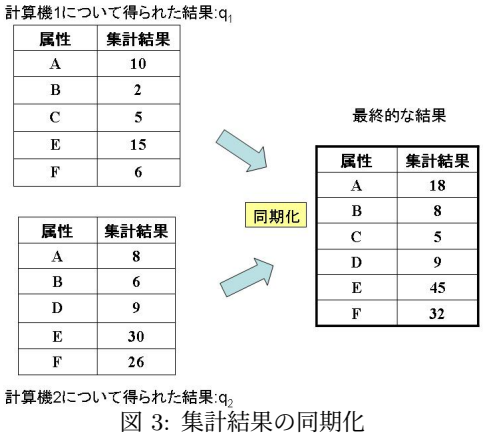

図 3: 集計結果の同期化

なるようにデータを分散させるために，上記の統一的な処理手順に沿って，各データ操作演算におけるデータ分散に必要な要件を述べる．

4. 並列化手法

前節で述べた HealthCube の処理手順の応答時間を，複数の計算機を使って短くするために，効率よく各計算機にデータを分散させることを考える．そこで，HealthCube の処理手順に沿って，各データ操作演算におけるデータ分散に必要な要件を述べる．

まず，区間選択演算 g では，各計算機になるべく均一にデータを分散させる必要がある．もし過度の偏りがあれば，データ量が多く保存されている計算機が区間選択をする際に負荷がかかってしまい，効率的でない．次に結合演算 β では，同一 ID のタプルの結合処理が行われるので，同一 ID のタプルが同一計算機に保存されている必要がある．もし，異なる計算機に同一 ID のタプルが分散されていた場合，タプルを結合するために計算機間の通信が必要になり，非効率である．最後に集計演算 α では，図 3 のように最後に各計算機で得られた結果を同期化させる必要がある．以上をまとめると，効率よくデータを分散させるためには次のような条件を持たなくてはならない．

- 各計算機のデータ量がなるべく均一で，かつ，同一 ID のタプルが同一計算機に保持されている．

この条件を満たすために，患者 ID をランダムに割り当てる．さらに，k 台の計算機に分散させる場合，テーブル INTERVAL I のタプルを，以下の条件を満たすように複数のテーブル I_i $(i = 1, \cdots, k)$ に分割し，各計算機に割り当てる．

- $I = \bigcup_{i=1,\cdots,k} I_i$
- $id(I_i) \cap id(I_j) = \emptyset \quad (i \neq j)$

ここで，$id(I)$ は，テーブル `INTERVAL` I の ID の集合を返す関数とする．1 つ目の条件は，テーブル `INTERVAL` I の情報が

図 4: I の分散の例

分割により失われないことを表している．また 2 つ目の条件により，同一 ID のタプルが異なる計算機に保存されていないので，上記の要件を満たす．例えば，$mod(ID, k)$ の値に応じて ID を分散させることで条件を満たす．具体的に図を用いて示すと，図 4 の上方の表のようにテーブル INTERVAL I のデータが保存されており，計算機数 $k = 2$ 台に INTERVAL I を分散させると図 4 の下表のように各計算機にはタプルの数がほぼ同じで，同一 ID のタプルも異なる計算機にまたがっていないように分散させることが出来る．さらに式 (1) のクエリは並列化によって次式になる．

$$\boldsymbol{\chi}(\bigcup_{\boldsymbol{i=1,\cdots,k}} \boldsymbol{\alpha(\beta(\{g(T, I_i, c_1), g(T, I_i, c_2), \cdots\}, O, W), \sigma)} \tag{3}$$

ここで，I_i はテーブル INTERVAL I を各計算機に分散させたときのテーブルである．χ の演算について，2 台の計算機にデータを分散した場合を例に説明する．テーブル I_1，I_2 それぞれで得られた結果を q_1，q_2 とすると，χ は図 3 のように q_1，q_2 の属性値が同じ集計結果について足し合わせる演算である．この χ の演算により最終結果を導くことが出来る．なお，q_i は $\alpha(\beta(\{g(T, I_i, c_1), g(T, I_i, c_2), \cdots\}, O, W), \sigma)$ であり，各計算機で独立に導き出されるため，q_i を得るまでに計算機間の通信は発生しない．χ の演算により計算機間の通信が発生するが，テーブル I_i に比べ q_i は小さいので，通信負荷が小さい．

以上より，データを分散させることが出来れば，各計算機がそれぞれ並列に前節で述べた処理手順を踏み，χ の演算により集計結果を同期化させることで，並列化していないときと同じの結果を得ることが出来る．

5. 並列化に対する理論的考察

第 2 節の HealthCube の基本的処理手順の中で，結合演算 β と集計演算 α の処理が，計算時間を要すると考えられる．そこで，本節では結合演算 β と集計演算 α の計算量を考察することで，並列化により理論的にはどの程度速度が向上するのかについて考察する．

5.1 結合演算 β における計算量の比較

一般に，リレーショナルデータベース管理システム (RDBMS) で結合演算 $\bowtie$ がどのように処理されるかを説明する [Mishra 1992]．RDBMS は，テーブル R_1 のタプル数 M とテーブル R_2 のタプル数 N に応じて，3 種類の方法を用いて結合処理 $R_1 \bowtie R_2$ をする場合が多い．以下に 3 種の結合と，M, N がどのような時に用いられるかを示す．

- ハッシュ結合 (HSJOIN): M, N の一方が多く，他方が少ない場合に用いられることが多い．
- ソート/マージ結合 (MSJOIN): $M \times N$ が大きく，検索条件によってテーブルの大部分が結合される場合に用いられることが多い．
- ネステッドループ結合 (NLJOIN): $M \times N$ が少なく，検索条件によってテーブルの多くを絞り込んだ結合をする場合に用いられることが多い．

ここで，HealthCube の区間選択演算 g によって得られたテーブルの集合 $\{I'_1, I'_2; \cdots, I'_n\}$ が存在するとき, 結合演算 β の処理を行う際に上の 3 つの結合方法が用いられる．HealthCube は大規模な医療データへの適用を想定しており，様々な絞込みを行う．絞込みを行う選択演算 g により出力される区間テーブル の 1 つは，比較的タプル数が少ない．従って，多くのクエリで適用される HSJOIN について，詳しく説明し，並列化している場合と並列化していない場合の計算量を比較する．

5.1.1 HSJOIN

従来の HealthCube において，分析者のクエリ $\alpha(\beta(\{g(T, I_1, c_i), g(T, I_1, c_j)\}, O, W), \sigma)$ が発行されたとき，区間選択演算 g によりタプル数に偏りがある区間情報 R と S(R のタプル数：$r \ll S$ のタプル数 : s) が選択されたとする．この場合，結合演算 β は，R の属性である ID と S の属性である ID が等しいときに結合を行うが，その結合は RDBMS において次のように処理される．

1. タプル数の少ないテーブル R の ID の値，またはその索引をハッシュキーとしたハッシュテーブルを作成する．
2. もう一方のテーブル S の ID の値，またはその索引をハッシュ関数にかけ，テーブル R と結合できるかを確認する．
3. ハッシュ値が等しいレコードを結合して結果を返す．

手順 2 はハッシュテーブルの検索に等しいので，テーブル S の 1 タプルに要する計算量が $O(1)$ となるために，全体としてテーブル S のタプル数分の計算量が必要となり，$O(s)$ となる．このような手順により，$O(s)$ の計算量で結合演算 β は処理される．

一方，提案手法を用いて，テーブル INTERVAL I を k 台 $(PC_1, PC_2, \cdots, PC_k)$ の計算機に分散させた並列環境にあるとする．先程と同じのクエリが発行されると，計算機 PC_i では区間選択演算 g によって，偏りのある区間情報 R_i，S_i が選択される．このときタプル数はそれぞれ，$r/k, s/k$ となる．そして，結合演算 β は前述の手順を踏み処理される．従って，PC_i は $O(s/k)$ の計算量で結合演算 β を処理することができる．他の計算機も同様に，結合演算 β は $O(s/k)$ で並列的に処理する．結果的に，並列化したときの結合演算 β は，従来の HealthCube と比べて $1/k$ の計算量で処理することができる．

5.2 集計演算 α における計算量の比較

集計演算 α の計算量は, σ によって異なるが, 本稿では σ の中で使用頻度も多く，計算量も一番多い $\sigma = count(distinct\ \ id)$ について考える．$count(distinct\ \ id)$ は結合演算 β で得た結果を ID ごとに集計する演算であるが，集計演算 α によって集計されるテーブルのタプル数と ID の異なり数を比べると，一般に後者が遥かに少ないので，RDBMS では ID 別にハッシュテーブルを用意し，集計対象のテーブルを 1 タプルずつハッシュ関数にかけて集計をすると考えられる．つまり，前述の HSJOIN の手順 1，2 を実行していると考えられる．した

がって，従来の HealthCube と k 台の計算機で並列化した場合を比較すると，並列化した方が $1/k$ の計算量で処理することができる．

結合演算 β，集計演算 α について計算量を考えることで，従来の HealthCube と比べて並列した場合 $1/k$ の計算量で処理することができる．つまり，理論的には k 倍速い処理 (線形速度向上) が可能である．

6. 提案手法の実装と評価実験

本研究では，提案した並列化の手法を Java を用いて実装した．各計算機で並列に処理を行う過程は，Java のマルチスレッドプログラミングを用いて実装した．マルチスレッドプログラミングとは，プロセス内の実行の流れであるスレッドを複数生成し，非常に短時間でスレッドを切り替えることによってほぼ並列的に処理を行うことのできるプログラム方法である．また，集計した結果を同期化させる手法は，1 つの n 次元配列の更新作業を行うことで実現した．

実データのデータ分布に従って生成された人工データを用いて，従来の HealthCube と 2 台，3 台の計算機にそれぞれデータを分散し並列化した HealthCube のクエリに対する応答時間を比較した. クエリの種類によって，HealthCube の処理手順が異なることから，クエリを以下の 4 つのタイプに分けて実験を行った．ここで，$g^{(1)}$, $g^{(2)}$ の違いを例を用いて述べると，$g^{(1)}$ は図 2 の術式の情報と CT 検査の情報の値で選択されるが，$g^{(2)}$ はそれぞれの値が OLAP のロールアップをした値で選択される．本稿では，実験 3 の結果について述べる．

- 実験 1: $\boldsymbol{\alpha(\beta(g^{(1)}(T,I,c_1)),\text{`}count(distinct\ id)\text{'})}$
- 実験 2: $\boldsymbol{\alpha(\beta(g^{(2)}(T,I,c_1)),\text{`}count(distinct\ id)\text{'})}$
- 実験 3: $\boldsymbol{\alpha(\beta(\{g^{(1)}(T,I,c_1),g^{(1)}(T,I,c_2)\}),\ \text{`}count(distinct\ id)\text{'})}$
- 実験 4: $\boldsymbol{\alpha(\beta(\{g^{(2)}(T,I,c_1),g^{(1)}(T,I,c_2)\}),\ \text{`}count(distinct\ id)\text{'})}$

実験 3 では，カテゴリ c_1 を固定し，カテゴリ c_2 を変化させることで評価実験を行った．図 5 は従来の HealthCube と並列化した HealthCube の応答時間を計測した散布図である．このグラフの横軸である対象件数は，区間選択演算 g により選択され，固定したカテゴリ c_2 と結合演算 β をして得られた件数である．また，図 6 は，提案手法が計測時間について，並列化していない場合よりもどれほど早く応答するかを示した図である．縦軸は 従来の応答時間/並列化した応答時間 の値で，横軸は図 5 と同様である．図 6 において，2 台，3 台の実験結果の双方で対象件数が少ないとき，速度の向上が k 倍以上になることが少なかったが，応答時間自体が非常に短いので分析者にとって不都合はないと考えられる．

7. まとめ

本研究ではデータを 2 台，3 台にそれぞれ分散させ，並列化を行った．その結果，速度向上率は対象件数が多くなるにつれて徐々に k 倍になることが確認できた．このように，本研究の提案手法によって理論的に応答時間が $1/k$ になることを示し，様々な実験を通して確認をすることができた．今後の方針として，より複雑なクエリを用いて実験を行うことで性能確認をすることが望まれる．

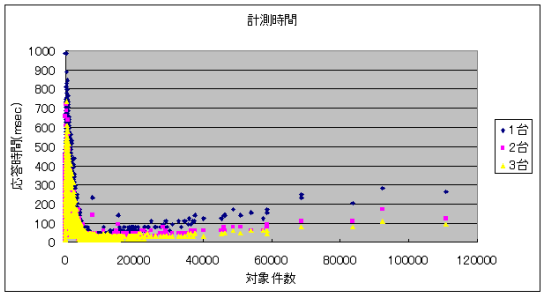

図 5: 計測時間 (実験 3)

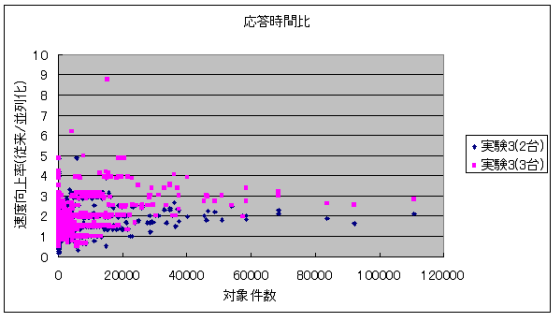

図 6: 速度向上率 (実験 3)

参考文献

[Mishra 1992] P. Mishra and M. Eich. Join Processing in Relational Databases. *ACM Computing Surveys*, Vol.24, No.1, pp.63-113, 1992.

[Pedersen 1999] T. Pedersen and C. Jensen. Multidimensional Data Modeling for Complex Data. *Proc. of Int'l Conf. on Data Engineering*, pp. 336-345, 1999.

[Chen 2004] Y. Chen, F. Dehne, T. Eavis and A. Chaplin. Parallel ROLAP Data Cube Construction on Shared-Nothing Multiprocessors. *Distributed and Parallel Databases*, Vol.15, No.3, pp.219-236, 2004.

[Wang 2005] L. Wang, A. Zhang, and M. Ramanathan. BioStar Models of Clinical and Genomic Data for Biomedical Data Warehouse Design. *Int'l Journal of Bioinformatics Research and Applications*, Vol.1, No.1, pp.63-80, 2005.

[Inokuchi 2006] A. Inokuchi and K. Takeda. A Method for Online Analytical Processing of Text Data. *Proc. of ACM Conf. on Information and Knowledge Management*, pp.455-464, 2006.

[高林 2007] 高林, 猪口, 鷲尾, 紀ノ定. 臨床プロセス解析を支援する OLAP システムの考察. データベースと Web 情報システムに関するシンポジウム (dbWeb2007), 7A-3, 2007.