5P-8

データベース専用オペレーティングシステムμOPT-Rにおける演算処理方式について

水谷 高康　国枝 和雄　大久保 英嗣　津田 孝夫
(京都大学工学部)

1. はじめに

近年、ハードウェア技術の向上やパーソナルコンピュータの普及によって、パーソナルコンピュータ上にデータベースシステム(DBMS)が構築されるようになってきている。しかし、パーソナルコンピュータでは、主記憶や2次記憶の容量に制限があるため、I/O処理時間の全体の処理時間に占める割合が極端に高くなっている。膨大なデータを処理するというDBMSの性質上、このことは重要な問題となる。また、現在実現されているパーソナルDBMSは汎用オペレーティングシステム(OS)上に構築されているものが多く、これらの汎用OSはデータベース処理に最適な環境を提供しているとは言い難い。我々は、以上の問題を解決するために、データベース処理に目的を限定したパーソナルデータベース専用OSμOPT-Rの開発を行っている。本稿では、μOPT-Rで採用している関係演算処理方式について述べる。

2. μOPT-Rの特徴

μOPT-Rは以下の特徴を備えている。

(1) マルチタスク機能の実現

従来のパーソナルコンピュータ用のOSではサポートされていないCPU処理とI/O処理のオーバラップを可能にするマルチタスク機能を標準で提供している。

(2) 2レベルのトランザクションスケジューリング

トランザクションを単位とした大域的スケジューリングと、トランザクション内での並行実行可能性を最大限に引き出すための局所的スケジューリングを行っている。

(3) ストリーム指向型方式による関係演算の高速処理

本稿で述べる関係演算処理方式であり、先に述べたマルチタスク機能を活用することによって、ほとんどの場合I/O処理時間をCPU処理時間の中に埋没させることが可能となっている。

(4) ソフトウェアセグメンテーションによるメモリ管理

この仮想記憶方式の実現により、アプリケーションプログラムは処理対象となるセグメントの名前だけを意識すればよい。

(5) 階層転置型ファイルを用いたデータベース構造

μOPT-Rでは、データベース構造として関係を属性単位に格納する階層転置型ファイルを用いており、これによって演算の実行時にアクセスを行うデータ量を大幅に削減することができる。

(6) 高い移植性を持ったシステム記述

μOPT-Rは、ハードウェアに依存する部分を除いてすべてC言語で記述されており、高い移植性を有している。

本関係演算処理方式ではこれらの機能を利用して、限られた主記憶容量のもとで高速な演算処理を実現している。

3. 関係演算処理方式

ユーザの入力したトランザクションは内部的には関係演算の系列に分解される。この系列は、2分木を構成し、木の個々のノードが1つの関係演算や論理演算に対応している。μOPT-Rではこれらの各ノードに1つのタスクを割り当てている。これをマルチタスク環境で実行することによってCPU処理とI/O処理のオーバラップを行い、演算処理の高速化を実現している。

3.1 演算の駆動方式

演算木に分解された関係演算の駆動方式には逐次駆動型、要求駆動型、データ駆動型が考えられる。以下にこれら3つの方式について説明する。

(1) 逐次駆動型方式

各ノードの演算の実行を制御フロー順序づけ(control flow ordering)[1]に基づいて行う方式で、実行を開始する前から静的に実行順序が決められている。処理の単位を分割することはなく、データ全体を処理する。つまり、そのノードの演算をすべて終了するまでは結果のデータを出力しない。この方式は、各ノードにタスクを割り当てるのではなく、各ノードを関数あるいは手続きとすることにより実現可能である。

(2) 要求駆動型方式

この方式においては、演算の実行順序はデータフロー順序づけ(data flow ordering)[1]によって決定される。すなわち、各ノードは入力データがそろうと実行可能になるが、要求が伝わっていないノードつ

A Processing Scheme for Relational Operations in Database Operating System μOPT-R

TAKAYASU MIZUTANI, KAZUO KUNIEDA, EIJI OKUBO and TAKAO TSUDA (Kyoto University)

まり駆動されていないノードは入力データがそろっても処理は開始されない。この方式の処理においては、最初に演算木の根に該当するノードが要求を出し、それが下位ノードへと次々に伝播していく。そして、要求が葉ノードに到着すると演算を開始して出力データが上位ノードへ次々と渡されていく。

(3) データ駆動型方式

要求駆動型方式と同様に、演算の実行順序はデータフロー順序づけに基づいており、入力データがそろった段階で処理が開始される。すなわち、要求が演算を駆動するのではなく、データが駆動することになる。当然、この場合(2)で述べたような要求を伝播することによるオーバヘッドは存在しない。

3.2 ストリーム指向型関係演算処理方式

ストリーム指向型関係演算処理方式は要求駆動型もしくはデータ駆動型方式によって各演算ノードが処理を進めていく演算処理方式である[2]。つまり、各演算ノードは演算結果を1ページ生成すると、それが最後の出力データか否かを示すフラグとともにデータが格納されているセグメント名を親ノードに送信する。この出力データを生成順に並べた系列をストリームと呼ぶ。μOPT-Rでは、関係を構成するタプルのタプル識別子がストリームの各データ要素に対応し、その系列がストリームとなる。図1にその概略を示す。

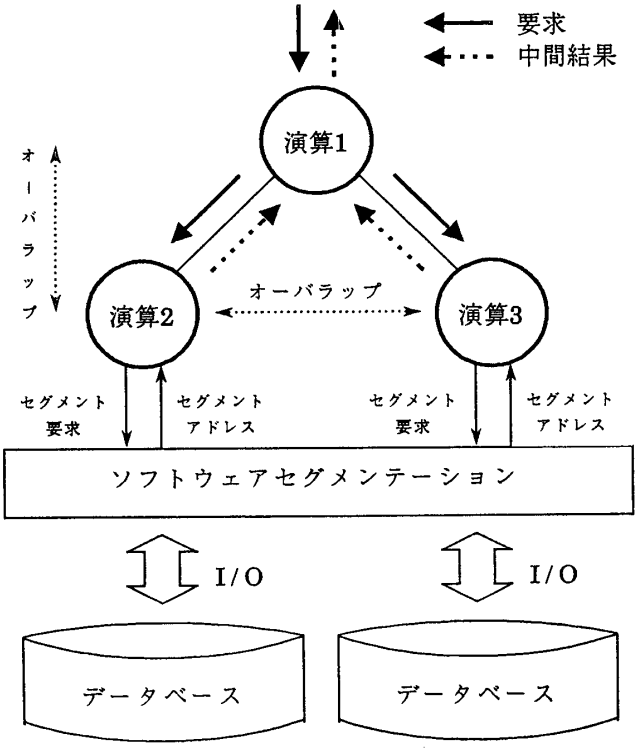

図1 μOPT-Rにおける関係演算処理方式

このように演算を行う単位をストリームに分割することによって、親ノードは、演算対象データの生産者である子ノードが演算を完了するのを待つことなく、演算の実行を開始することができる。つまり、本関係演算処理方式では、演算木レベルにおいて次の2種類の並列(あるいは並行)実行可能性を検出していることになる。

(1) 水平方向の並列性(兄弟ノード間の並列性)
(2) 垂直方向の並列性(親子ノード間の並列性)

これにより、CPU処理時間とI/O処理時間の多種多様なオーバラップを実現することが可能となり、CPUのアイドル時間を可能な限り短くすることができる。したがって、パーソナルDBMSにおける最大の問題点であるI/O処理時間をCPU処理時間の中に埋め込む可能性が高くなり、関係演算の高速処理を実現することができる。また、演算結果はその演算がまだ終了していなくても演算木の上位ノードの方へ次々に渡されていくので、システムの応答時間は大幅に短縮される。

一方、ストリーム指向型方式では各演算ノード内に大量の中間データを残すことになる。これは、2項関係演算が、演算に関与する関係の全ページを相互に比較しなければならないためである。つまり、どちらか一方の演算が完了していない場合、もう一方の入力データを捨てることができないことになる。マルチプロセッサシステムにおいては、この問題を解決する方法としてインナーリレーションの再計算によるアルゴリズムが見いだされている[2]。しかし、この方法をシングルプロセッサシステムであるμOPT-Rに用いるとインナーリレーションの再計算によるオーバヘッドが膨大なものになってしまう。μOPT-Rではソフトウェアセグメンテーションを実現しているため、一時的関係がメモリに入りきらない場合はそのセグメントが自動的に2次記憶へページアウトされる。従って、中間データを格納する場合のメモリ不足は発生し得ない。この場合I/O回数は増えるが、インナーリレーションを再計算するよりは短い時間で処理できる。

4. おわりに

本稿では、μOPT-Rで用いているストリーム指向型関係演算処理方式について述べてきた。本関係演算処理方式は、演算の処理対象をデータのストリームとしてとらえることによって、より細かいレベルでCPU処理時間とI/O処理時間をオーバラップさせることを目的としている。μOPT-Rでは、ソフトウェアセグメンテーションによる仮想記憶方式を採用しているので、演算の中間結果の蓄積によるメモリ不足といった問題に対処することが可能である。本関係演算処理方式は、CPU処理速度とI/O処理速度の格差が非常に大きいパーソナルコンピュータでは有効な演算処理方式と言えよう。

(参考文献)

[1] J. A. Sharp: Data Flow Computing, Ellis Horwood Ltd., (1985).

[2] 清木康, 長谷川隆三, 雨宮真人: 先行・遅延評価を用いた関係演算処理方式, 情報処理学会論文誌, Vol. 26, No. 4, 685-695 (1985).