Kappa-P のトランザクション制御

5R−6

椿野宣行[1], 佐藤裕幸[2], 河村元夫[3]

1: 三菱電機東部コンピュータシステム(株) 2: 三菱電機(株)情報電子研究所
3: (財)新世代コンピュータ技術開発機構

1 はじめに

Kappa-P[1]は、ICOTで研究開発された並列推論マシンPIM[2]上で動作する並列データベース管理システムである。Kappa-Pの構成は、複数の独立したデータベース管理システム(ローカルDBMS)を一つのデータベース管理システムとするいわゆる分散データベース管理システムの構成となっている。このローカルDBMSはPIMの一つのクラスタに割り当てられている。

本システムは疎結合並列と密結合並列の両方を考慮している。この中のトランザクションは、ユーザから見た一つのトランザクションを複数のローカルDBMSのサブトランザクションの集まりとすることで疎結合並列処理を実現している。またローカルDBMSはクラスタに割り当てられるので一つのローカルDBMS内の個々のトランザクションもまた並列に実行される。

本発表では、Kappa-Pが用いた密結合並列を考慮した同時実行制御および一貫性の保証のためコミットメント制御について述べる。

2 トランザクション管理の概要

トランザクションとは一般に複数のI／Oから構成される分割できないひとまとまりの処理であり、一貫性の保証のため同時実行制御とコミットメント制御が必要である。Kappa-Pでは、トランザクション管理として次のような同時実行制御とコミットメント制御をおこなっている。

2.1 同時実行制御

同時実行制御については、知識情報処理の環境ではタプル単位等の細かい制御の必要性があまりなかったこと、同じ資源(テーブル)に対する使用要求の数はそれほど多くないことなどの理由により、テーブル単位の排他制御のみおこなうこととした。すなわち、トランザクション開始時に、read_onlyかexclusiveでテーブルをロックし、トランザクション終了時にアンロックするというものである。この時問題となるのは、テーブルの施錠の管理方法である。Kappa-Pの構成だと全体もしくはローカルDBMS単位での集中管理が考えられるが、その場合実行時にボトルネックになり並列性が阻害される可能性がある。このため各テーブル自身が管理を行なう分散ロック方式を用いている。

2.2 コミットメント制御

トランザクションのコミットメント制御は、1トランザクションが複数ローカルDBMSのサブトランザクションの集まりからなるため、分散コミットメントを行う必要がある。分散コミットメントには、2相コミット[3]やブロック状態を減らすための3相コミット[3]などが知られている。対象であるPIMは、分散システムではなく並列マシン(しかもプロトタイプ)であり、3相コミットによりブロック状態を減らす必要性は大きくなかった。このため分散システムとしての一貫性の保証に重点をおき、分散コミットメントを最低限保証するプロトコルとして、2相コミットプロトコルの中で最も単純なものを採用した。

3 分散ロック

テーブルの施錠管理を各テーブル自身がおこなう分散ロック方式では、ボトルネックが解消できる反面デッドロックの検出とその解消が問題となってくる。このデッドロックの問題をKappa-Pでは、時間監視により検出しリトライを行なうことで解消することとしている。Kappa-Pでのトランザクションでは分散ロックを次のように行なっている。

3.1 テーブルのロック

ロック要求は各サブトランザクションが受け付け、各テーブルにロック要求を出す。各テーブルはキューを持ちロック要求を受け付ける。すなわちアンロック状態になったらキューの先頭の要求がロックできたこととなる。しかしある時間待って全てのテーブルがロックできない場合は、ロック要求を取り消しリトライ(3.2)をおこなう。トランザクションは全てのサブトランザクションでのロックが完了してから開始可能となる。またトランザクション開始後は新たにロックできない。

水平分割テーブルに関しては、構成テーブルを一度に全てロックしようとするとデッドロックになる可能性が高いので、代表テーブルを決めて最初はそれのみをロックし、その後で残りの全テーブルをまとめてロックするようにしている。

Transaction Management on Kappa-P
Noriyuki TSUBAKINO[1], Hiroyuki SATO[2], Moto KAWAMURA[3]
1:Mitsubishi Electric Computer System(Tokyo) Corporation 2:Mitsubishi Electric Corporation 3:Institute for New Generation Computer Technology.

3.2 リトライ

サブトランザクションの中で一つでも開始できないサブトランザクションがあった場合は、開始できた全てのサブトランザクションを無効にした後ランダムな時間待ってリトライを行なう。ランダムな時間にすることで同一テーブルをロックしようとした他のトランザクションとのデッドロックを避けている。

またリトライする場合は、デッドロック監視時間を少し増やすことにより、多くのテーブルを使用するトランザクションが不利にならないようにしている。

4 2相コミットプロトコル

一貫性の保証とできるだけ並列性を阻害しないためテーブルをロックしている時間をできるだけ短くすることを考慮している。トランザクションは以下の順で実行される。

(1) テーブルの分散ロック
サブトランザクションを生成しテーブルのロックを行なう。

(2) トランザクションの開始
複数のサブトランザクションを取りまとめるためのサブトランザクション（コーディネータ）を一つ決定する。これがトランザクションの開始宣言でもある。コーディネータは、全サブトランザクションの監視、各サブトランザクションへのコマンドの振り分けなどを行なう。

(3) コマンドの実行
テーブルをアクセスするコマンドには検索系, 読み系, 更新系などがあり全てコーディネータが一括して受け付ける。受け付けたコマンドは各サブトランザクションに渡され実行される。コマンドの形式は以下の通りである。

コマンド：{サブコマンド, サブコマンド, ...}
サブコマンド：各サブトランザクションに対するコマンドである。

(4) トランザクションの終了
コーディネータは、2相コミットプロトコルに従って全サブトランザクションをコミットする。

4.1 コマンドのシリアライズ

コーディネータは各サブトランザクションに対するコマンドを一括して受けとり各サブトランザクションに渡している。ここで問題になるのがコマンドの追い越しである。例えば検索コマンドの実行後更新コマンドを実行する場合などである。このためにコマンドのシリアライズを行なう必要がある。これはコーディネータで行なえば簡単であるが、コーディネータが受けとった一つのコマンドの中で最も処理時間の長いサブコマンドの実行が終るまで待つのは効率が悪いので各サブトランザクションでおこなっている。

4.2 エラー処理

エラー監視はコーディネータと各サブトランザクションがそれぞれおこなっている。コーディネータは受け付けた全コマンドのステータスを監視する。各サブトランザクションは自身で受け付けたコマンドのステータスを監視する。

サブトランザクションのなかで一つでもエラーが発生すれば、それはコーディネータが発見する。コーディネータは全サブトランザクションにアボート通知を出し全ての処理を中断させた後全トランザクションのアボートを行なう。エラーが発生したトランザクションはそれ自身でアボートを行なう。

4.3 障害回復処理

1相目のコミット後コーディネータダウンもしくは通信エラーなどによりサブトランザクション自身コミットかアボートかが不明の場合、他の全サブトランザクションにどうしていいか尋ねる。その結果の一つでもコミットかアボートかの返事があればそれを行ないサブトランザクションを終了する。これによってコーディネータが立ち上がらなくてもコミットかアボートかが判断できる場合は、資源の解放(テーブルのアンロック)が可能となる。それ以外の場合はコーディネータの指示を待つ。

コミットかアボートかが不明のままシステムが落された場合は、システム立ち上げ時に障害が発生した時と同じ状態を再現し上記と同じ処理を行なう。立ち上げ処理と障害回復処理は並列に実行されるため、障害回復処理の終了を待たずに立ち上がることができる。

5 まとめ

疎結合並列マシン PIM/m のプロセッサ数 64 で水平分割テーブルを用いた評価をおこなった。この結果トランザクションへ発行された実行コマンドやコミット処理での制御コマンドは、全サブトランザクションにほとんど同時に到達し各サブトランザクションの処理がほぼ同時に開始されていることが確認された。これは分散データベースの手法が有効であったと言える。

今後は密結合並列マシン PIM/p での評価をおこない密結合並列での有効性を検証したい。

参考文献

[1] 河村ほか: 並列データベース管理システム Kappa-P の概要, 第45回情処全国大会, 5R-3, 1992-10.
[2] 瀧： "Parallel Inference Machine PIM" The International Conference on Fifth Generation Computer Systems '92, 1992.
[3] P.A.Bernstein, V.Hadzilacos, N.Goodman, "Concurrency Controland Recovery in Database Systems", Addison-wesley, 1987.