並列／分散システムにおけるトランザクション処理の並列性制御自動選択方式

廣上雅久　大澤範高　弓場敏嗣
電気通信大学大学院情報システム学研究科
〒182 東京都調布市調布ヶ丘 1-5-1
E-mail: {hirokami, osawa, yuba}@yuba.is.uec.ac.jp
http://www.yuba.is.uec.ac.jp/

本論文では、並列／分散システムにおけるトランザクション処理において、トランザクションの状態を検出して適宜自動的に並列性制御方式を変更する「並列性制御自動選択方式」を提案する。並列性制御方式は、大きくロッキング方式・時刻印方式・楽観的方式に分けられ、それぞれ長所・短所を持つ。従来の多くのシステムにおいてはCamelot/Avalon、Argusなどのように並列性制御方式は固定であるが、固定しない場合でもIXIなどのようにプログラマが選択する方式が一般的である。また本論文では、この「並列性制御自動選択方式」を分散記憶型並列計算機Cenju-3上で実装し、トランザクション処理におけるスループットの向上の評価を行い提案方式の効果を検証する。

An Automatic Concurrency Control of Transaction Processing on Parallel and Distributed Systems

Norihisa HIROKAMI, Noritaka OSAWA and Toshitsugu YUBA,
Graduate School of Information Systems,
The University of Electro-Communications
1-5-1, Chofugaoka, Chofu, Tokyo 182, JAPAN
E-mail: {hirokami, osawa, yuba}@yuba.is.uec.ac.jp
http://www.yuba.is.uec.ac.jp/

In this paper, we propose a new method of an "automatic" concurrency control for parallel and distributed transaction processing systems. There are three types of concurrency control methods; a locking method, a time stamp ordering method and an optimistic method. It is well known these methods have some advantages and some disadvantages as well, depending on workload status of a transaction processing system. The new method changes automatically its concurrency control method into the most effective one, according to the workload information of transactions in operation. We have implemented a prototype of a parallel and distributed transaction processing system on the parallel computer Cenju-3. The effectiveness of the proposed method is quantitatively evaluated by using the prototype.

1 はじめに

トランザクション処理は、複数のデータの読み書きなどの操作を一括して処理する方法で、銀行の口座管理、列車の座席予約などのデータベースシステムに利用されている。近年では、単一の汎用大型コンピュータの下でのトランザクション処理から並列/分散システム化が進んでおり、マルチメディア化に伴い扱うデータも多種多様になっている。

トランザクション処理は、一般的にACID特性[1]を保たねばならないという特徴がある。ACID特性は、原子性(Atomicity)、一貫性(Consistency)、分離性(Isolation)、持続性(Durability)からなる。トランザクション処理におけるACID特性を保つ手段として、分散多重化制御、エラー回復制御、並列性制御などがある。本論文では並列/分散システムにおける並列性制御に着目した。並列性制御は、トランザクションの並列実行が直列実行した結果と等価に、すなわち直列化可能でなければいけないため、何らかの規則によってデータのアクセスを制限する。

並列/分散システムにおけるトランザクション処理において並列性制御は、ロッキング方式[2]、時刻印方式[3]、楽観的方式[4]に大別される。これらの並列性制御の各方式はそれぞれ利点と欠点を持ち、あらゆる状況で最適な並列性制御方式は存在しない。そこで、状況に応じて最適な並列性制御方式を自動的に選択するのが望ましいと考える。本論文では、トランザクションの性質を示すデータを収集し、その内容に応じて適宜自動的に並列性制御方式を変更する「並列性制御自動選択方式」を提案する。また同方式を並列/分散システムの環境においてCライブラリで実現し、トランザクションのスループットの向上を評価検証する。

2 並列性制御方式と既存システム

ロッキング方式[2]は、データアクセス前にデータにロックをかけ、アクセスが終了した時点でアンロックを行う方法である。この方式では、デッドロックが発生する可能性があり、この防止、回避機構などは並列性を大きく損なう。また、読み出しのみのトランザクションなどのように、ロッキングが実際には必要ではない割合が極めて高く、データ競合が少ない場合は無駄である。時刻印方式[3]はトランザクション生成時の論理時刻印で並列性制御を行う方法である。この方法では、デッドロックは発生しないが、順序を乱すトランザクションがあると、理論的には何ら問題がないにもかかわらずアボートが発生する場合がある。楽観的方式[4]は、2つ以上のトランザクションが同時に同一データにアクセスする可能性が少ない仮定で、トランザクションの実行、競合の検証、反映(コミットまたはアボートを行う)の3つの処理手順からなる。この方式では、ロールバックのオーバヘッドが大きいため、データ競合が多い場合はこの方法は向かない場合がある。

既存の並列/分散トランザクション処理システムにおいては、MITのArgus[5]やカーネギメロン大学のCamelot/Avalon[6]などは、ロッキング方式で並列性制御を行う。また京都大学のIXI[7]は、適応型時刻印方式と呼ばれる3つの方式(予約型時刻印方式、排他ロック式時刻印方式多重版時刻印方式)からプログラマが並列性制御方式を選択する方式を採用している。

3 並列性制御自動選択方式

並列性制御自動選択方式は、トランザクションの性質を示すデータを収集し、その内容に応じて適宜自動的に並列性制御方式を選択する方法である。データ競合が起こる確率α、デッドロックの起こる確率β、ロールバックが起こる確率γにより、図1のように状態が遷移する。

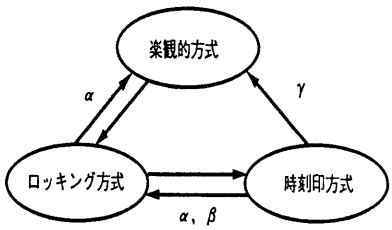

図1: 並列性制御自動選択方式状態遷移図

楽観的方式・ロッキング方式・時刻印方式におけるトランザクション1件あたりの実行時間T_o[s], T_l[s], T_t[s]は、式(1)(2)(3)で表される。ここで、逐次状態でのトランザクション1件の平均処理時間をt[s]、単位時間当たりの平均実行トランザクション件数をn[件]、楽観的方式・ロッキング方式・時刻印方式における1件当たりのオーバヘッド時間をそれぞれt_o[s], t_l[s], t_t[s]、タイムアウト時間をT_{out}とする。

$$T_o(\alpha) = \frac{(t+t_o)}{n(1-\alpha)} \quad (1)$$

$$T_l(\alpha,\beta) = \frac{(t+t_l)(1+\alpha)}{n} + \frac{\beta\, T_{out}}{n(1-\beta)} \quad (2)$$

$$T_t(\gamma) = \frac{(t+t_t)}{n(1-\gamma)} = \frac{t+t_t}{n-n\alpha+1} \quad (3)$$

式(3)において、タイムスタンプ方式では、競合が起きた場合でも確実に1件は処理される。よって、$\gamma = \alpha - 1/n$ となる。

次に図2のように、楽観的方式からロッキング方式への自動切替を考える。ここで、α が α_1 から α_2 に増加したとき並列性制御方式を図の実線のように切替えることにより、斜線部分の速度向上が期待できる。楽観的方式からロッキング方式に切替えるオーバヘッド時間を T_{ol} として、

$$T_o(\alpha) > T_l(\alpha,\beta) + T_{ol} \quad (4)$$

で自動切替の効果が現れる。この場合、デッドロックは考えないので、$\beta = 0$ とする。よって式(6)で求めた α を切替の閾値とする。

$$T_o(\alpha) = T_l(\alpha,0) + T_{ol} \quad (5)$$

$$\alpha = \frac{-nT_{ol} \pm \sqrt{n^2T_{ol}^2 - 4(t+t_l)(t_o - t_l - nT_{ol})}}{2(t+t_l)} \quad (6)$$

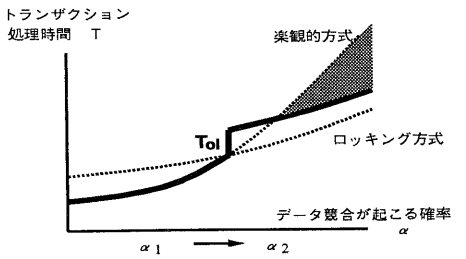

図2: 楽観的方式からロッキング方式への切替効果

同様にロッキング方式から楽観的方式への自動切替の場合は、図3で示すように、切替オーバヘッド時間を T_{lo} として、

$$T_l(\alpha,0) > T_o(\alpha) + T_{lo} \quad (7)$$

で自動切替の効果が現れる。よって、式(9)で求めた α を切替の閾値とする。

$$T_l(\alpha,0) = T_o(\alpha) + T_{lo} \quad (8)$$

$$\alpha = \frac{-nT_{lo} \pm \sqrt{n^2T_{lo}^2 + 4(t+t_l)(t_o - t_l + nT_{lo})}}{2(t+t_l)} \quad (9)$$

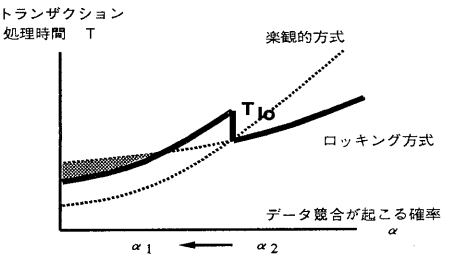

図3: ロッキング方式から楽観的方式への切替効果

ロッキング方式から時刻印方式への自動切替の閾値は、切替オーバーヘッド時間を T_{lt} として、

$$T_l(\alpha,\beta) > T_t(\alpha) + T_{lt} \quad (10)$$

を満たす α, β で自動切替の効果が現れる。また、時刻印方式からロッキング方式への自動切替では切替オーバヘッド時間を T_{tl} として、

$$T_t(\alpha) > T_l(\alpha,\beta) + T_{tl} \quad (11)$$

を満たす場合、自動切替の効果が現れる。時刻印方式から楽観的方式への自動切替では、切替オーバヘッド時間を T_{to} として、

$$T_t(\alpha) > T_o(\alpha) + T_{to} \quad (12)$$

で自動切替の効果が現れる。

4 並列/分散システムへの実装

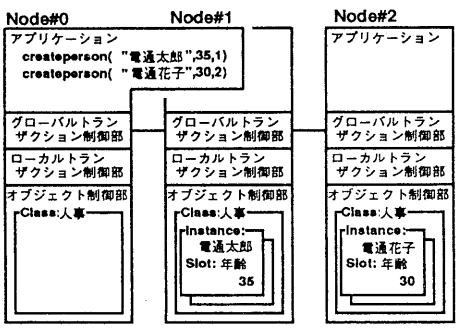

図4: 試作システムの動作

「並列性制御自動選択方式」を分散記憶型並列/分散システムに実装した。図4で示すように、試作並列/分散トランザクションシステムはC言語のライブラリ形式で実現され、グローバルトランザクション制御部、ローカルトランザクション制御部、オブジェクト制御部からなり、アプリケーションプ

ログラムから利用される。また、自動切替のパラメータである α, β, γ は、グローバルトランザクション制御部において、モニタリングされている。また、ノード間の通信手段としてメッセージパッシング方式を用いており、リクエスト/リプライのプロトコルで通信を行う。

このシステムの特徴としては、弱い意味でのオブジェクト指向データを取り扱うことができる。これはデータオブジェクトとしてクラスを定義でき、それによってクラスの配下にインスタンスを生成できる。また、クラスにはスロットを定義でき、インスタンスのスロットにデータを格納できる構造になっている。図 4は、createperson というトランザクションがアプリケーションプログラムから生成される例で、人名のインスタンスを生成し、スロット年齢に値を代入する。

次に、試作並列/分散トランザクション処理システムにおける各並列性制御方式の動作について考える。図 5に楽観的方式、図 6にロッキング方式、図 7に時刻印方式におけるメッセージの流れを示す。

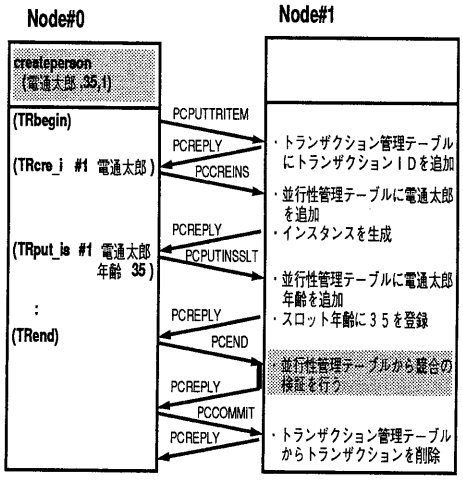

図 5: 楽観的方式におけるメッセージの流れ

図において、網で囲まれた部分が各方式の独特な処理部分である。これによると、ロッキング方式や時刻印方式では、1 件の処理ごとに並行性管理テーブルを検索し競合を検証しているが、楽観的方式においては 1 件の処理ごとでは並行性管理テーブルにデータを追加するだけで、最後のコミット処理で競合の検証を行い、競合がなければそのトランザクションはコミットされ、競合があればアボートしロールバックさせる。

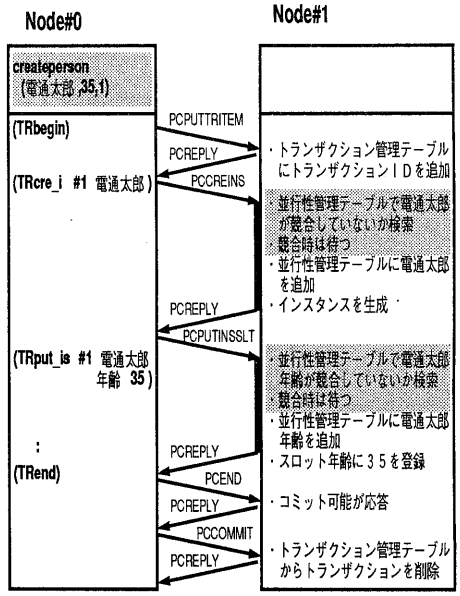

図 6: ロッキング方式におけるメッセージの流れ

またロッキング方式においては、競合がある場合データオブジェクトへのアクセスは待たされるが、時刻印方式においては時刻印を検索することにより、コミットかアボートの判断が下される。これらの背景からも、競合が少ないときは楽観的方式がスループットが速く、競合が多いときはロッキング方式や時刻印方式がスループットが速いことがわかる。

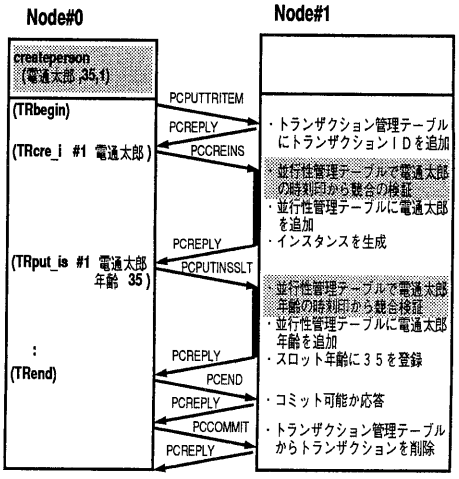

図 7: 時刻印方式におけるメッセージの流れ

5 評価検証

試作した並列/分散トランザクション処理システムを用いて、テストトランザクションプログラムを実行し、分散記憶型並列計算機 Cenju-3 上で「並列性制御自動選択方式」の有効性を評価検証した。

［実験 1］競合がない場合のトランザクション処理時間の測定

競合がない場合のトランザクション処理時間を測定し、競合がない場合は楽観的方式が一番効率がよいことを検証した。ここで、インスタンススロットの参照を 5 回行うトランザクションを用い、16 台の PE において一定時間間隔でトランザクションを実行した。

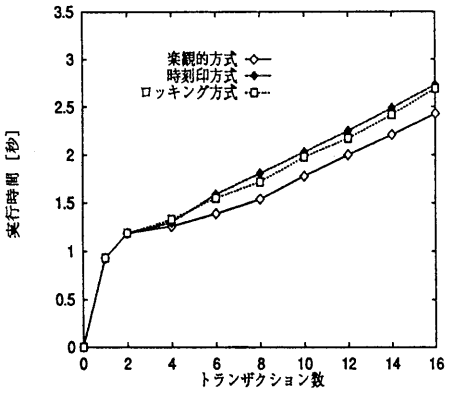

図 8: Cenju-3 における競合がない場合のトランザクション処理時間の測定

［実験 2］楽観的方式からロッキング方式への自動切替効果の測定

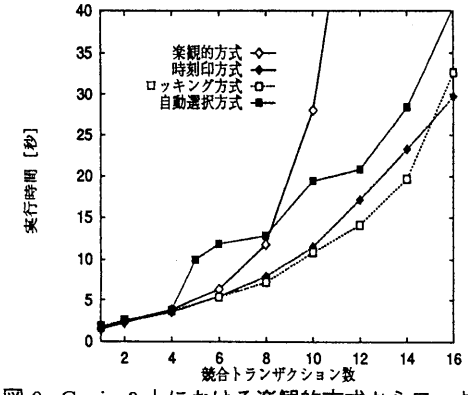

図 9: Cenju-3 上における楽観的方式からロッキング方式への自動切替効果の測定

競合トランザクション数が増加傾向にある場合のトランザクション処理時間を測定した。この場合、並列性制御自動選択方式により、競合トランザクション数が 4 超えた時に楽観的方式からロッキング方式に切替えた時の効果を検証する。ここで、インスタンススロットの参照を 5 回、更新を 1 回行うトランザクションを用い、16 台の PE において一定時間間隔でトランザクションを実行した。

［実験 3］ロッキング方式から楽観的方式への自動切替効果の測定

競合トランザクション数が減少傾向にある場合のトランザクション処理時間を測定した。この場合、並列性制御自動選択方式により、競合トランザクション数が 4 より小さくなった時にロッキング方式から楽観的方式に切替えた時の効果を検証する。ここで、インスタンススロットの参照を 5 回、更新を 1 回行うトランザクションを用い、16 台の PE において一定時間間隔でトランザクションを実行した。

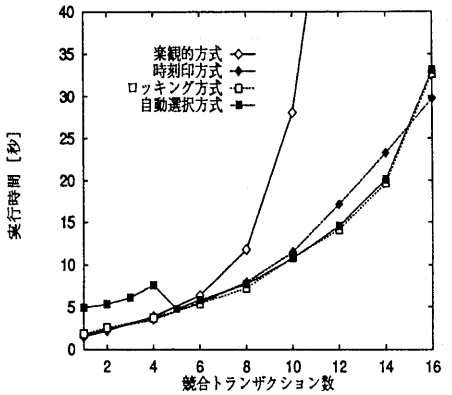

図 10: Cenju-3 上におけるロッキング方式から楽観的方式への自動切替効果の測定

楽観的方式、ロッキング方式、時刻印方式におけるメッセージのプロトコルを図 5、図 6、図 7にそれぞれ示したが、これからも明らかなように、ロッキング方式や時刻印方式では、1 件の処理ごとに並行性管理テーブルを検索し競合を検証しているのに対して、楽観的方式においては 1 件の処理ごとでは並行性管理テーブルにデータを追加するだけで、最後のコミット処理で競合の検証を行い、競合がなければそのトランザクションはコミットされ、競合があればアボートしロールバックする。

「並列性制御自動選択方式」においては、競合がない時は楽観的方式で並列性制御を行うが、実験 1

の結果からも楽観的方式が一番速いことが実証された。競合が殆んどないトランザクションの実行においては、ロッキング方式で行うよりも「並列性制御自動選択方式」で選択された楽観的方式が有効である。

競合トランザクション数が増加傾向にある場合は、実験2において並列性制御を「並列性制御自動選択方式」により楽観的方式からロッキング方式へ切替えた。この例では、競合トランザクション数が4を超えた時点で、トランザクションの受付を中止し、現在動作しているすべてのトランザクションがコミットまたはアボートすると、並列性制御方式を変更し、トランザクションの受付を再開する。これによると、一時的に切替えオーバヘッドで楽観的方式よりスループットが悪くなるが、競合トランザクション数が8を超えると大きく効果が現れた。競合トランザクション数が10の場合で28.6 %、12の場合で78.0 %の速度向上がある。しかし競合トランザクション数が4から8までは、切替える前の並列性制御方式である楽観的方式よりも効率が悪い。競合トランザクション数が6の時で71 %悪化する。これを回避させるためには、切替えオーバヘッドを減らすことや、通信メッセージを減らすなどの工夫が必要である。

競合トランザクション数が減少傾向にある場合は、実験3において並列性制御を「並列性制御自動選択方式」によりロッキング方式から楽観的方式へ切替えた。この例では、競合トランザクション数が4以下では楽観的方式が他方式より若干速いが、切替えオーバヘッドが大きく効果は見られなかった。しかし、トランザクションの競合が4以下で長時間連続する状態においては、1度だけの切替え時間は無視できる。また、実験1にあるように、競合がない場合にはそのままのロッキング方式より、楽観的方式の方が効率がよいことから「並列性制御自動選択方式」で選択された楽観的方式が有効である。

6 おわりに

本稿では、「並列性制御自動選択方式」を提案し、分散記憶型並列計算機Cenju-3上でMPI(Message Passing Interface)を用い、C言語のライブラリとして実装した。また、このシステムを用いて、トランザクションのスループットの向上を評価し、提案の方式の効果の検証を行った。

「並列性制御自動選択方式」は、トランザクションの性質を示すデータから、随時、データ競合、デッドロック、ロールバックが起こる確率を計算し、これらが閾値を超えた場合、自動的にロッキング方式、時刻印方式、楽観的方式の中から並列性制御方式を変更するモデルである。試作したシステムは、10,000ステップを越える大規模なものになった。また、このシステムを利用して、提案の方式の効果の検証を行ない、「並列性制御自動選択方式」は有効であることを示した。

トランザクション処理分野においては、並列/分散化が進んでおり、またマルチメディア化に伴い扱うデータも多種多様になっている。試作システムの特徴は、並列/分散システムをベースとして、弱い意味でのオブジェクト指向データを取り扱うことができ、これらに対応している。

今後の課題としては、「並列性制御自動選択方式」の問題として、デッドロックやロールバックが同時に多発するトランザクションを実行したとき、並列性制御方式の自動切替の多発や振動などの問題が発生する可能性がある。これらを未然に防ぐ工夫も必要である。また、超並列や超分散に対応するため、通信メッセージを減らすなどの工夫に取り組み、並列／分散トランザクション処理システムとしての完成を目指す必要がある。

謝辞 分散記憶型並列計算機Cenju-3の利用環境をご提供頂いた、NEC C&C研究所 並列処理センタに感謝致します。

参考文献

[1] Haerder, T., Reuter, A.: "Principles of Transaction-Oriented Database Recovery," ACM Computing Surveys, Vol.15, No.4 (1983).

[2] Eswaran, K.P.: "The Notions of Consistency and Predicate Locks in a Database System," CACM, Vol.19, No.11, pp.624-633 (1976).

[3] Thomas, R.H.: "A Majority Consensus Approach to Concurrency Control for Multiple Copy Database," ACM Trans. on Database Systems, Vol.4, No.2, pp.180-209 (1979).

[4] Kung, H.T.: "On Optimistic Methods for Concurrency Control," ACM Trans. on Database Systems, Vol.6, No.2, pp.213-226 (1981).

[5] Liskov, B., Curtis, D., Johnson, P., Scheifler, R.: "Implementation of Argus," Proceedings of 11th ACM Symposium on Operating Systems Principles, pp.111-122 (1987).

[6] Eppinger, J.L., Mummert, L.B. Spector, A.Z.: *Camelot and Avalon - A Distributed Transaction Facility*, Morgan Kaufman Publishers (1991).

[7] 國枝和雄, 各務達人, 大久保英嗣, 津田孝夫: "分散トランザクションシステム IXI," 情報処理学会誌, Vol.35, No.6, pp.1185-1199 (1994).