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

3L－5

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†

1　はじめに

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[1]、時刻印方式[2]、楽観的方式[3]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[1]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[2]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[3]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[4]やカーネギメロン大学のCamelot/Avalon[5]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[6]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。

†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, {hirokami,osawa,yuba}@yuba.is.uec.ac.jp

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。

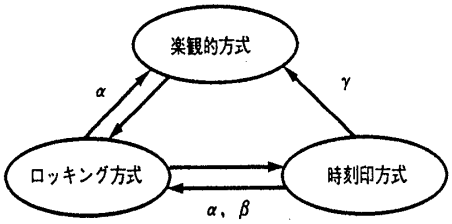

図1: 並列性制御自動選択方式状態遷移図

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。

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

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

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

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$T_o(\alpha) > T_l(\alpha,\beta) + T_{ol}$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(4)で求めたαを切替の閾値とする。

$$\frac{(t+t_o)}{n(1-\alpha)} = \frac{(t+t_l)(1+\alpha)+T_{out}}{n} + T_{ol} \quad (4)$$

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、切替オーバヘッド時間をT_{lo}として

$T_l(\alpha, 0) > T_o(\alpha) + T_{ol}$で自動切替の効果が現れる。よって、式(5)で求めたαを切替の閾値とする。

$$\frac{(t+t_l)(1+\alpha)+T_{out}}{n} = \frac{(t+t_o)}{n(1-\alpha)} + T_{lo} \qquad (5)$$

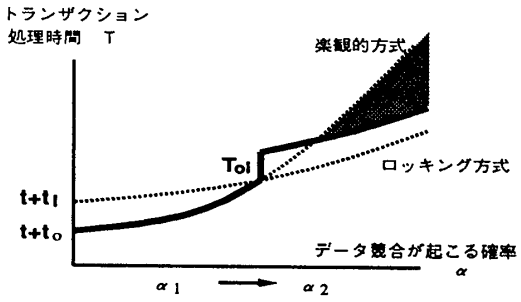

図 2: 楽観的方式からロッキング方式への切替効果

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、$T_l(\alpha, \beta) > T_t(\alpha) + T_{lt}$を満たす$\alpha$, 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、$T_t(\alpha) > T_l(\alpha, \beta) + T_{tl}$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、$T_t(\alpha) > T_o(\alpha) + T_{to}$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

「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3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。

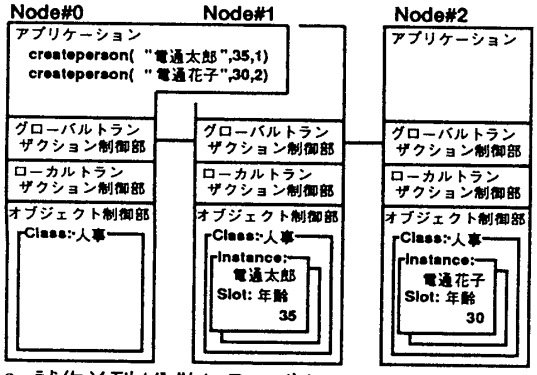

図 3: 試作並列/分散トランザクションシステムの動作

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3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。

5 評価実験

分散記憶型並列計算機 Cenju-3 上でインスタンススロットの参照5回、更新1回行うトランザクションを用いて、各並列性制御方式の実行時間を測定した。この例では競合トランザクション数が4を超えると楽観的方式においてロールバックが多発し効率が悪くなる。また、競合トランザクション数が6を超えるとロッキング方式が一番効率が良いことがわかる。よって、競合トランザクション数が4を超えたときに楽観的方式からロッキング方式に切替えることにより効果が現れる。

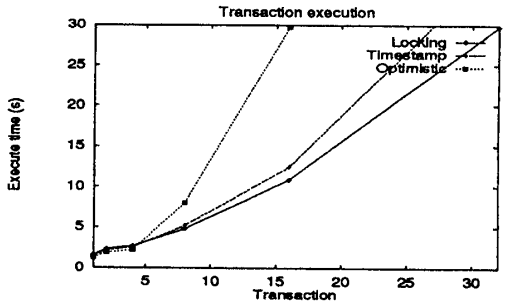

図 4: Cenju-3 上での評価実験

6 おわりに

本稿では、「並列性制御自動選択方式」について述べた。並列/分散システムにおける試作システムの実験から、競合が多い場合はロッキング方式や時刻印方式が効率がよく、少ない場合ではわずかではあるが楽観的方式が効率がよいことがわかった。現在、「並列性制御自動選択方式」を実装評価中であり、今後いろいろなパターンのトランザクションを用いて有効性を評価検証する予定である。

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

[2] 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).

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

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

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

[6] 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).