5AA－7

並行処理制御における独立化可能クラスの有効性検証
— 施錠方式に対するシミュレーション —

†権藤 裕二　†徐 海燕　‡史 一華　††古川 哲也
†福岡工業大学　‡福岡工業短期大学　††九州大学

1　まえがき

データベースにおける並行処理制御は、直列可能性基準に基づいて行なわれてきている。しかし、協同作業を必要としたり長時間の処理を含む高水準データベースでは、直列可能性を満足しなければならないことによる問題点が広く認識されている。そのため、一貫性情報を利用することにより正当なクラスを独立化可能クラスに拡大する研究が行われている[2]。独立化可能クラスは、論理性を満たし、かつ一貫したデータベースから検索を行い、変更された結果も一貫している処理単位からなるスケジュールと等価なスケジュールからなる。本稿では、独立化可能基準の有効性を確認するために、それぞれの基準の施錠方式に対して行なったシミュレーションの結果を報告する。

2　独立化可能性に基づく施錠方式

個々の利用者のデータベースに対する操作は、処理単位Tという全順序関係$<$を持つ検索操作$(R(X))$、変更操作$(W(X))$の集合で表される。ここで、$R(X)$, $W(X)$はXに属するすべての要素に対する同時検索、変更操作である。複数の処理単位（処理単位集合$\boldsymbol{T}$）の並行実行をスケジュールHと呼ぶ。

直列可能性に基づく施錠方式の中で最もよく知られているのは２相施錠方式である。

[定義 1] 各$R_i(X) \in H$または$W_i(X) \in H$を行なう前にX中の各要素に対して共有施錠または専有施錠を行ない、かつ各処理単位$T_i \in \boldsymbol{T}$は施錠部分と解錠部分とに分かれる並行処理制御方式を、２相施錠（2PL）方式という。ただし、すでに専有施錠されているデータ項目を他の処理単位が再び施錠することはなく、すでに共有施錠されているデータ項目を他の処理単位が再び専有施錠することはない。□

一貫性情報を利用できると、並行処理制御の正当性基準は次の独立化可能性基準まで拡張できる[2]。

独立化可能なスケジュールのクラスは、各処理単位T_iが次の３つの性質を満たす等価なスケジュールが存在するものからなる[2]。

検索一貫性： 処理単位T_iによって検索されるのは、ある一貫したデータベースの部分集合である。

結果一貫性： 検索されたデータベースに対して、単独実行した結果のデータベースは一貫している。

論理性： 処理単位T_iによって操作されたデータ項目は、T_iが終了するまで、他の処理単位$T_j(j \neq i)$によって変更されていない。

スケジュールHの独立化可能性を判定する問題は、一貫性に関する問題と並行処理制御に関する問題に分けられ、後者に対しては次のような施錠方式がある[2]。

[定義 2] 各$R_i(X) \in H$または$W_i(X) \in H$を行なう前にX中の各要素に対して共有施錠または専有施錠を行ない、かつ各処理単位$T_i \in \boldsymbol{T}$は施錠部分と解錠部分とに分かれる並行処理制御方式を、拡張２相施錠（2PLE）方式という。ただし、すでに施錠されているデータ項目を他の処理単位が再び専有施錠することはない。□

2PLE 方式に従うスケジュールが独立化可能である[2]。2PL 方式と比較すると、2PLE 方式はすでに専有施錠されているデータ項目には再び共有施錠できるという拡張がなされている。

3　シミュレーションによる比較

定性的には、2PLE 方式は 2PL 方式での共有施錠と専有施錠、専有施錠と共有施錠、専有施錠同士という３つの競合の中の専有施錠と共有施錠間の競合関係をなくしている。2PL と 2PLE 方式の差を定量的に評価するために、本稿では、論文[1]の性能評価モデルをもとに仮想データベースを構築し、シミュレーションを行なった。その設定値は、データベースのサイズを 1000、端末の数を 200 とし、MPL（同時実行可能な処理単位数）を 200 まで変化させ、10 秒間に処理できた処理単位の総数（スループット）を測定し比較検討した。各端末から生成される処理単位は平均で 8 回の検索操作と 2 回の変更操作を行なう。処理単位中のすべての操作が終了すれば、1 sec の外部思考時間の後に、再び新しい操作を持った処理単位として開始される。

Efficiency of the Equivalent-Independent Class in Concurrency Control – Simulations of Locking Protocols
Yuuji GONDOU†, Haiyan XU†,
Yihua SHI‡, Tetsuya FURUKAWA††
†Fukuoka Institute of Technology,
††Fukuoka Junior College of Technology,
‡Kyushu University

すべてのシミュレーションは同じスケジュールから開始している。処理単位が開始されると、まずシステム内で実行されている処理単位の数をチェックし、すでにMPLだけの数が実行されているならその処理単位は待機列(ready queue)に入り、実行中の処理単位が終了するか後退復帰されるのを待つ。MPLだけの数が実行されていなければその処理単位は並行処理列(cc queue)に入り、並行処理制御要請を出す。許可されれば、検索・変更操作を行なう。許可されなければ、処理単位は待ち列(block queue)に入る。この待ち列の中ですくみが発生したら、最後に待ち列に入った処理単位が後退復帰される。またこのシミュレーションでは同じプログラムを20回実行させその平均によって性能評価をした。次に、主なシミュレーション結果について述べる。

「スループットによる比較」 図1は、CPU数を25、DISK数を50としたときのシミュレーション結果である。MPLが小さい時には2PLと2PLE方式の差はあまり見られないが、MPLが増加するにつれてその差は大きくなり、最大で47%もの差が見られた。

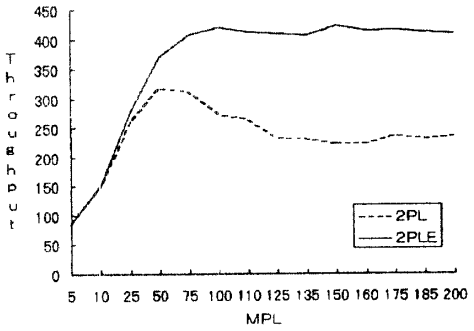

図1　有限資源（CPU数25, DISK数50）における2PLと2PLE方式の比較

「ブロック率と後退復帰率による比較」 実行される操作中の何割がblock queueに入るかを示すブロック率と、実行される処理単位中の何割が後退復帰になるかを示す後退復帰率について、2PLと2PLE方式を比較してみた。ブロック率は2PLE方式が2PL方式の三分の二位であり、後退復帰率は2PLE方式の方が若干低いという結果になっている。

「応答時間とその標準偏差による比較」 各処理単位の処理開始から終了までかかる平均の応答時間とその標準偏差を両方式について比較してみた。応答時間は2PLE方式の方が短いという結果になっている。スループットに関する結果と合わせると、これは2PLE方式の方が効率良く処理が実行されており、並行性が高いことを物語っている。標準偏差は2PLE方式の方が倍ぐらい小さくなっている。これは2PLE方式の方がブロックされる可能性が低いためと考えられる。ただし、実行状況の分析より、最初に実行された処理単位が最後まで残ってしまうことがある結果が得られている。これは変更操作がブロックされている間に、そのデータ項目に対する検索操作は次々と実行できる2PLE方式の特徴を示している。

「資源数による比較」 資源を限りあるものから限りないものまで変化させてシミュレーションを行なった。資源に限りがある場合のCPUとDISKの数は1:2となるように設定し、CPUの数を1, 5, 10, 25, 50, 無限のように変化させた。無限とは、処理の要請があればいつでもCPUやDISKが使用できるような環境にあることを意味する。CPUの数が1や2というように資源が少ない時には、2PLと2PLE方式の差はそれほど見られないが、資源が増加するに従って2PLと2PLE方式の差は次第に拡大し、CPUの数が10の場合から明確な差が見られるようになる。図1はCPUの数が25の場合の結果であるが、無限資源の場合には、資源によって操作が待たされることがないので、図1の結果に比べて2PLと2PLE方式の差はさらに15%程度拡大した。

「処理単位の大きさによる比較」 1つの処理単位の平均検索操作数を16個、平均変更操作数を4個と処理単位のサイズを2倍にしたシミュレーションを行なった。MPLが小さい時に2PLと2PLE方式の差は24%になり今までのものの6%と比べその差が顕著になった。これは、処理単位のサイズが大きくなったことにより競合も増加したためである。競合が発生した時に、2PL方式ではすべての場合において処理単位が短い時よりも長時間待たなければならないが、2PLE方式では検索操作が長時間待つことがなく、それだけ多くの処理単位が実行できる。

「思考時間による比較」 データベースを会話方式で使用することを想定して、検索操作を行なった後1/2の割合で思考時間を入れてシミュレーションを行なった。2PLと2PLE方式との差は思考時間を入れる時と入れない時とではそれほど相違はない。思考時間を入れる時と入れない時とでは基本的に競合の発生する数は同じためである。

4　まとめ

本稿では、施錠方式に対するシミュレーションにより、独立化可能性という一貫性情報を持つ高水準データベースにおける新たな基準と、従来の直列可能基準を比較した。各場合とも並行性上で明確な差が見られ、独立化可能基準に基づく施錠方式が高水準データベースにおいて有効であることが確認できた。

参考文献

[1] Pakesh Agrawal: Concurrency Control Performance Modeling: Alternatives and Implications, ACM TODS, Vol. 12, No. 4, pp. 609-654 (1987).

[2] 徐　海燕, 古川哲也, 史　一華: 並行処理制御方式による独立化可能クラスと直列可能クラス比較, 情報処理学会論文誌, Vol. 35, No. 8, pp. 1600-1609 (1996).